



ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ

**ΤΜΗΜΑ
ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ**

JAVA SWING

ΚΕΡΑΜΟΠΟΥΛΟΣ ΕΥΚΛΕΙΔΗΣ

JAVA SWING – ΜΙΑ ΠΡΩΤΗ ΓΕΥΣΗ

Μία πρώτη μικρή γεύση των δυνατοτήτων της Swing.

Παράδειγμα Δυνατοτήτων Swing

JAVA SWING - ΕΠΙΓΡΑΜΜΑΤΙΚΑ

- ◉ Το JAVA SWING είναι ένα API της JAVA το οποίο χρησιμοποιείται για την κατασκευή Γραφικών Διεπαφών Χρήστη.
- ◉ Είναι κομμάτι του JFC (Java Foundation Classes)
- ◉ Μπορεί να θεωρηθεί ως εξέλιξη του AWT (Abstract Window Toolkit) της Java.
- ◉ Βασίζεται στο μοντέλο **MVC** (Model View Control)

JAVA FOUNDATION CLASSES

- ◎ Το JFC αποτελείται από 5 μέρη.
 1. Swing GUI Components
 2. Pluggable Look-and-Feel Support
 3. Accessibility API
 4. Java 2D API
 5. Internationalization

SWING GUI COMPONENTS

- ⦿ Περιέχει κλάσεις για γραφικά συστατικά από τα οποία αποτελείται μία γραφική διεπαφή.
- ⦿ Ειδικότερα, περιλαμβάνει κλασικά στοιχεία σύγχρονων γραφικών διεπαφών όπως,

Menu

ColorChooser

EditorPane

ProgressBar

SplitPane

ToolTips

Frames

Combobox

List

ScrollPane

TabbedPane

Tree

Buttons

FileChooser

OptionPane

Slider

Table

SWING – PLUGGABLE ΕΜΦΑΝΙΣΗ ΚΑΙ ΑΙΣΘΗΣΗ

- Με αυτό το χαρακτηριστικό δίνει την δυνατότητα στον προγραμματιστή να δημιουργήσει γραφική διεπαφή, όπου ο χρήστης θα μπορεί να ρυθμίσει την εμφάνιση και αίσθηση της διεπαφής σύμφωνα με τις δικές του προτιμήσεις.
- Τέσσερις μορφές είναι διαθέσιμες:
 - Metal (Java),
 - Windows (Microsoft),
 - Motif (XWindows) και
 - User Defined.

ACCESSIBILITY API

- ⦿ Περιέχει τάξεις και μεθόδους που κάνουν εύχρηστη τη διεπαφή για ανθρώπους με ειδικές ικανότητες.
- ⦿ Δίνει τη δυνατότητα χρήσης σχετικών συσκευών, όπως
 - Screen readers και
 - Braille.

JAVA 2D

- ⦿ Περιέχει τάξεις και μεθόδους για τη χρήση σε εφαρμογές και applets:
 - υψηλής ποιότητας γραφικών 2D,
 - κειμένου, και
 - εικόνων.
- ⦿ Δημιουργία υψηλής ποιότητας εκτυπώσεων.

INTERNATIONALIZATION

- ⦿ Διευκολύνει τη δημιουργία διεπαφών που απευθύνονται σε χρήστες από διαφορετικές χώρες και οι οποίοι χρησιμοποιούν διαφορετική γλώσσα.

AWT – Ο ΠΡΟΓΟΝΟΣ ΤΗΣ SWING

- Όταν δημιουργήθηκε η Java, το API AWT (Abstract Window Toolkit) χρησιμοποιούνταν για την δημιουργία GUI.
- Το AWT υποστηρίζει τάξεις οι οποίες υλοποιούν τα βασικά στοιχεία μιας διεπιφάνειας.
- Ο λόγος της ριζικής βελτίωσης του AWT με τη SWING ήταν ότι:
 - Η μετάφραση των οπτικών (visual) συστατικών του AWT ήταν εξαρτημένη από τη πλατφόρμα (λειτουργικό) όπου δημιουργούνταν.
 - Έτσι τα συστατικά του μεταφράζονταν σε native code και χαρακτηρίζονταν ως «βαριά» συστατικά.
- Αυτό οδηγούσε σε προβλήματα όπως:
 - Δεν μπορούσε να υλοποιηθεί το μοτο της JAVA (γράψε μία φορά, τρέξτο παντού).
 - Η αίσθηση των συστατικών ενός GUI όταν εκτελούνταν σε άλλο λειτουργικό δεν μπορούσε να αλλάξει εύκολα.
 - Τα «βαριά» συστατικά του AWT δεν ήταν ευέλικτα και συνήθως είχαν περιορισμούς στην εμφάνιση.

SWING – ΗΡΘΕ ΓΙΑ ΝΑ ΞΕΠΕΡΑΣΕΙ ΤΟΥΣ ΠΕΡΙΟΡΙΣΜΟΥΣ ΤΟΥ AWT

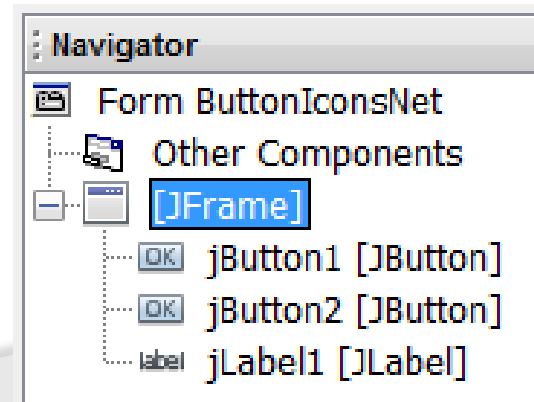
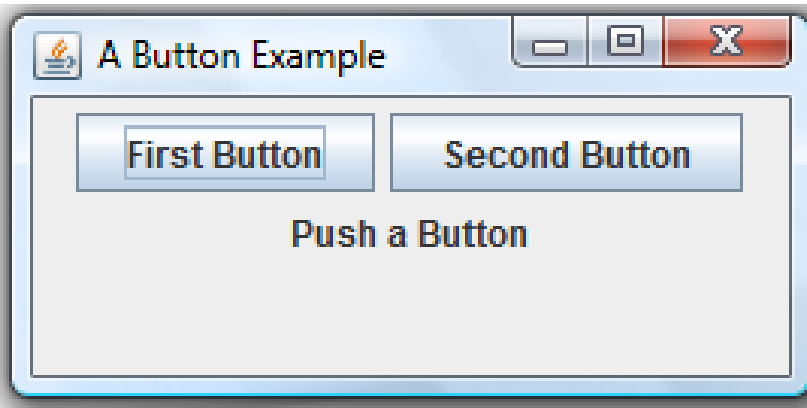
- ◎ Η Swing δημιουργήθηκε το 1997 ως ανάγκη για να λύσει τα προβλήματα του AWT.
- ◎ Δύο είναι τα κύρια χαρακτηριστικά της Swing
 - Ελαφριά συστατικά
 - Pluggable εμφάνιση και αίσθηση

SWING – ΕΛΑΦΡΑ ΣΥΣΤΑΤΙΚΑ

- ◎ Η μεγάλη πλειοψηφία των συστατικών της Swing είναι «ελαφριά»
 - Είναι γραμμένα αποκλειστικά σε Java.
- ◎ Αυτό είχε αρκετά πλεονεκτήματα
 - Τα συστατικά ήταν πιο ευέλικτα και πιο ευμετάβλητα (μπορούμε να αλλάξουμε εύκολα πολλές ιδιότητες τους).
 - Η εμφάνιση και αίσθηση των συστατικών ορίζεται μόνο από την Swing και όχι από την πλατφόρμα.

SWING – ΣΥΣΤΑΤΙΚΑ ΚΑΙ ΥΠΟΔΟΧΕΙΣ

- Τα συστατικά (components) είναι όλα τα στοιχεία που μπορεί να περιέχει μία διεπιφάνεια. Για παράδειγμα: κουμπιά, μενού, λίστες ...
- Οι υποδοχείς (containers) είναι και αυτοί συστατικά που περιέχουν άλλα συστατικά.
- Κάθε Swing εφαρμογή θα πρέπει να περιέχει έναν εξωτερικό υποδοχέα, ο οποίος θα περιλαμβάνει όλα τα συστατικά της διεπιφάνειας.
- Ένας υποδοχέας μπορεί να περιλαμβάνει και άλλους υποδοχείς.
- Μία διεπιφάνεια χτισμένη σε Swing μπορεί να παρασταθεί και ως ένα δέντρο όπου η ρίζα του είναι ο εξωτερικός υποδοχέας.



SWING – ΣΥΣΤΑΤΙΚΑ

- Όλα τα Swing συστατικά είναι παιδιά της τάξης **JComponent** εκτός από τέσσερα συστατικά-υποδοχείς
- Στην τάξη JComponent υλοποιούνται τα χαρακτηριστικά που είναι κοινά για όλα τα συστατικά μίας διεπιφάνειας, π.χ. η εμφάνιση και η αίσθηση.
- Όλα τα συστατικά περιέχονται στο πακέτο **javax.swing**.

SWING – ΥΠΟΔΟΧΕΙΣ

- ◎ Υπάρχουν δύο ειδών υποδοχείς:
 - Οι Εξωτερικοί,
 - Οι Εσωτερικοί

ΕΞΩΤΕΡΙΚΟΙ ΥΠΟΔΟΧΕΙΣ

- ◎ Είναι βαριά συστατικά και **δεν** είναι παιδιά της JComponent. Είναι τέσσερις
 - **JWindow** – Είναι ένα παράθυρο, το οποίο όμως δεν έχει την μπάρα στην κορυφή που περιέχει τον τίτλο του παραθύρου και τα κουμπιά για ελαχιστοποίηση, επαναφορά/μεγιστοποίηση και κλείσιμο.
 - **JFrame** – Το κλασικό παράθυρο για desktop εφαρμογές.
 - **JApplet** – Καταλαμβάνει ένα παραλληλόγραμμο στον browser, όπου αναπτύσσεται μία διεπιφάνεια σε Java Swing.
 - **JDialog** – Εξαρτημένο παράθυρο από κάποιο άλλο παράθυρο JWindow ή JFrame ή JApplet, ειδικό για διαχείριση διαλόγων.

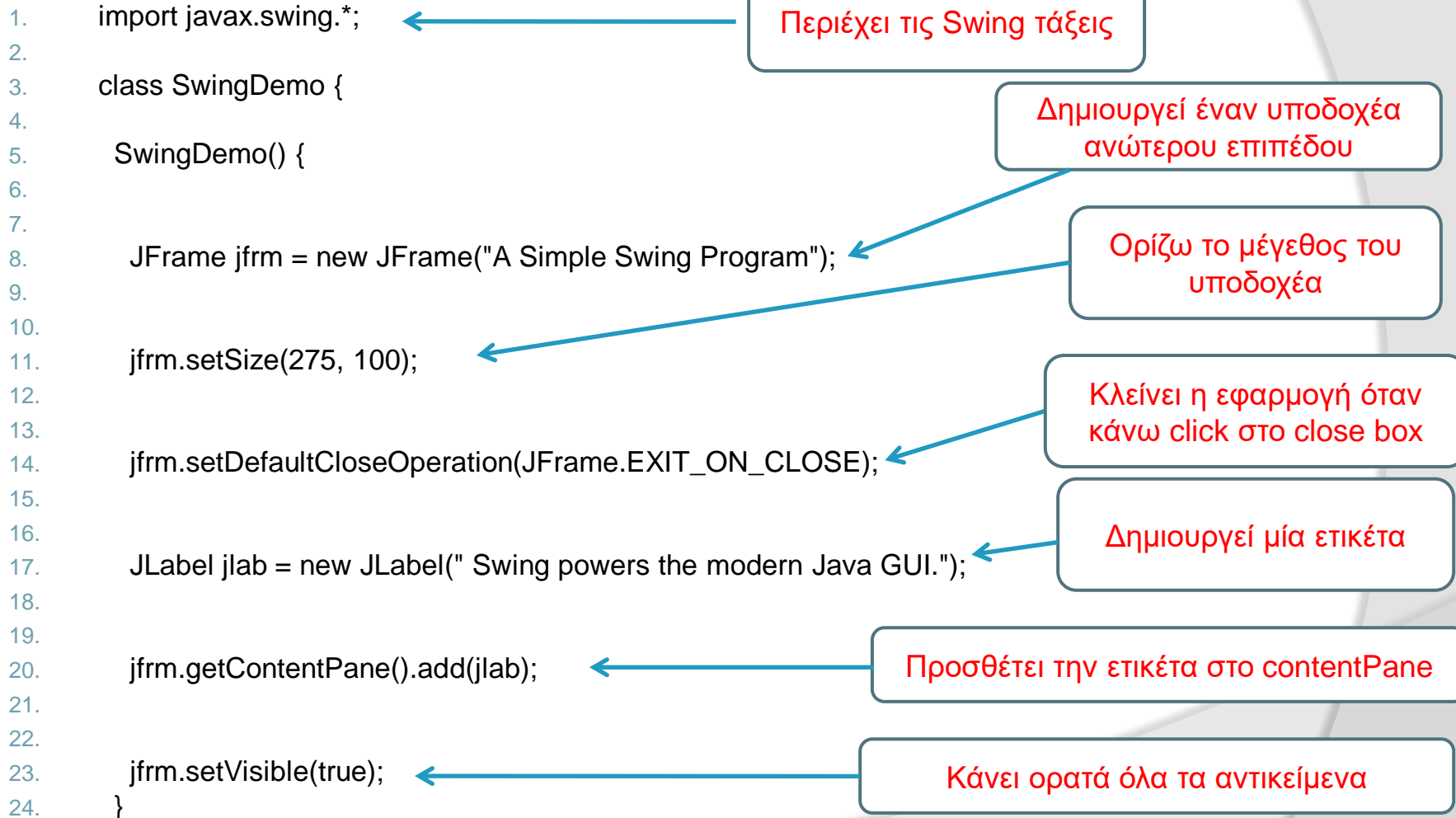
ΕΣΩΤΕΡΙΚΟΙ ΥΠΟΔΟΧΕΙΣ

- ⦿ Είναι ελαφριά συστατικά και είναι παιδιά της JComponent. Είναι δύο
 - JPanel
 - JRootPane
- ⦿ Χρησιμοποιούνται για να οργανώνουν σελ από συστατικά σε ομάδες.
- ⦿ Προφανώς οι εσωτερικοί υποδοχείς εισάγονται είτε σε κάποιο άλλο εξωτερικό ή εσωτερικό υποδοχέα.

ΥΠΟΔΟΧΕΙΣ ΤΟΥ ΑΝΩΤΕΡΟΥ ΕΠΙΠΕΔΟΥ

- Ο υποδοχέας του ανώτερου επιπέδου αποτελείται από αντικείμενο της τάξης JRootPane και ο ρόλος του είναι να περιέχει όλα τα συστατικά της εφαρμογής και ένα προαιρετικό μενού.
- Ο υποδοχέας ανώτερου επιπέδου αποτελείται από τρεις υποδοχείς:
 - **Glass pane**
Είναι διάφανος και ο ρόλος του είναι να διαχειρίζεται τα **γεγονότα** που δημιουργούνται από τις ενέργειες που προκαλούνται με το ποντίκι στο υποδοχέα ανώτερου επιπέδου και τα συστατικά του.
 - **Layered pane**
Διαχειρίζεται τις αλληλοεπικαλύψεις των συστατικών, με άλλα λόγια επιτρέπει τα συστατικά να τοποθετούνται σε **επίπεδα**.
 - **Content pane**
Οι δύο προηγούμενοι υποδοχείς χρησιμοποιούνται πίσω από τη «σκηνή». Ο υποδοχέας όπου ουσιαστικά προστίθενται όλα τα συστατικά είναι ο συγκεκριμένος.
- Μπορούμε να αναφερόμαστε στους τρεις υποδοχείς με τα ονόματα `glassPane`, `layeredPane`, `contentPane`

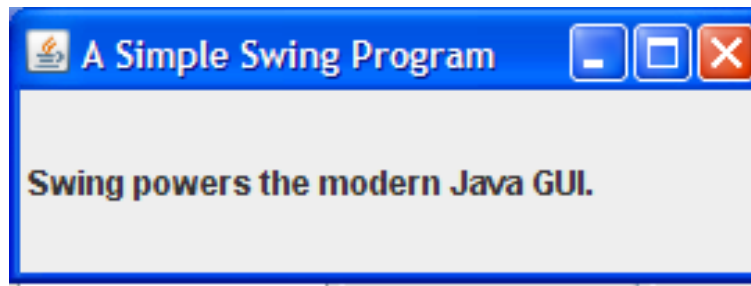
ΤΟ ΠΡΩΤΟ ΠΑΡΑΔΕΙΓΜΑ



ΤΟ ΠΡΩΤΟ ΠΑΡΑΔΕΙΓΜΑ

```
25. public static void main(String args[]) {  
26.  
27.  
28.     SwingUtilities.invokeLater(new Runnable() {  
29.         public void run() {  
30.             new SwingDemo();  
31.         }  
32.     });  
33.  
34. }  
35. }
```

← Δημιουργεί την Διεπιφάνεια



ΠΡΩΤΟ ΠΑΡΑΔΕΙΓΜΑ

ΣΥΣΤΑΤΙΚΑ ΠΟΥ ΧΡΗΣΙΜΟΠΟΙΗΘΗΚΑΝ

- JFrame είναι εξωτερικός υποδοχέας. Υλοποιεί ένα παράθυρο, με μπάρα τίτλου και κουμπιά για ελαχιστοποίηση, μεγιστοποίηση και κλείσιμο του παραθύρου.
- JLabel είναι ένα συστατικό της Swing το οποίο χρησιμοποιείται για να εμφανίζει πληροφορίες. Είναι το πιο απλό συστατικό αφού είναι παθητικό, δηλαδή δεν μπορεί να δεχθεί ενέργειες του χρήστη.

ΠΡΩΤΟ ΠΑΡΑΔΕΙΓΜΑ - ΕΠΕΞΗΓΗΣΗ

- Μέθοδος `setSize()`: ορίζει το μέγεθος του παράθυρου
`void setSize(int width, int height)`
- Μέθοδος `setDefaultCloseOperation()`: ορίζει τι θα συμβαίνει στην εφαρμογή όταν ο χρήστης επιλέγει να κλείσει ένα παράθυρο.
`void setDefaultCloseOperation(int what)`

Οι τιμές που μπορεί να πάρει το **what** είναι:

1. `EXIT_ON_CLOSE` : στην έξοδο σταματάει και η εφαρμογή
2. `DISPOSE_ON_CLOSE`: στην έξοδο σταματάει και η εφαρμογή
3. `HIDE_ON_CLOSE`: στην έξοδο η εφαρμογή δουλεύει στο φόντο
4. `DO_NOTHING_ON_CLOSE`: στην έξοδο να μην γίνει τίποτα

ΠΡΩΤΟ ΠΑΡΑΔΕΙΓΜΑ - ΕΠΕΞΗΓΗΣΗ

- Μέθοδος `add()`: προσθέτει ένα συστατικό στον υποδοχέα.
`Component add(Component comp)`
- Μέθοδος `getContentPane()`: επιστρέφει μία αναφορά με το `content pane` του υποδοχέα.
`Container getContentPane()`

Σημείωση:

Μετά το JDK5 η μέθοδος `getContentPane` παραλείπεται.

Δηλαδή, οι παρακάτω εντολές είναι ισοδύναμες

```
jfrm.add(jlab);
```

```
Jfrm.getContentPane().add(jlab);
```

ΠΡΩΤΟ ΠΑΡΑΔΕΙΓΜΑ - ΕΠΕΞΗΓΗΣΗ

- Μέθοδος `setVisible()`: εμφανίζει έναν υποδοχέα. Η αρχική κατάσταση είναι να είναι αόρατος ο υποδοχέας.

```
void setVisible(Boolean flag)
```

Οι τιμές της `flag` είναι `true` για ορατό και `false` για αόρατο.

- ```
SwingUtilities.invokeLater(new Runnable() {
 public void run() {
 new SwingDemo();
 }
});
```

Τα προγράμματα Swing είναι event-driven (προγραμματισμός γεγονότων) γι' αυτό και ενεργοποιούνται στο νήμα-γεγονότων και όχι στο νήμα-main. Με το αντικείμενο `Runnable` ενεργοποιούμε το νήμα-γεγονότων.

Τις διεπιφάνειες τις ενεργοποιούμε πάνω στο νήμα-γεγονότων με τις μεθόδους

```
static invokeLater(Runnable obj)
```

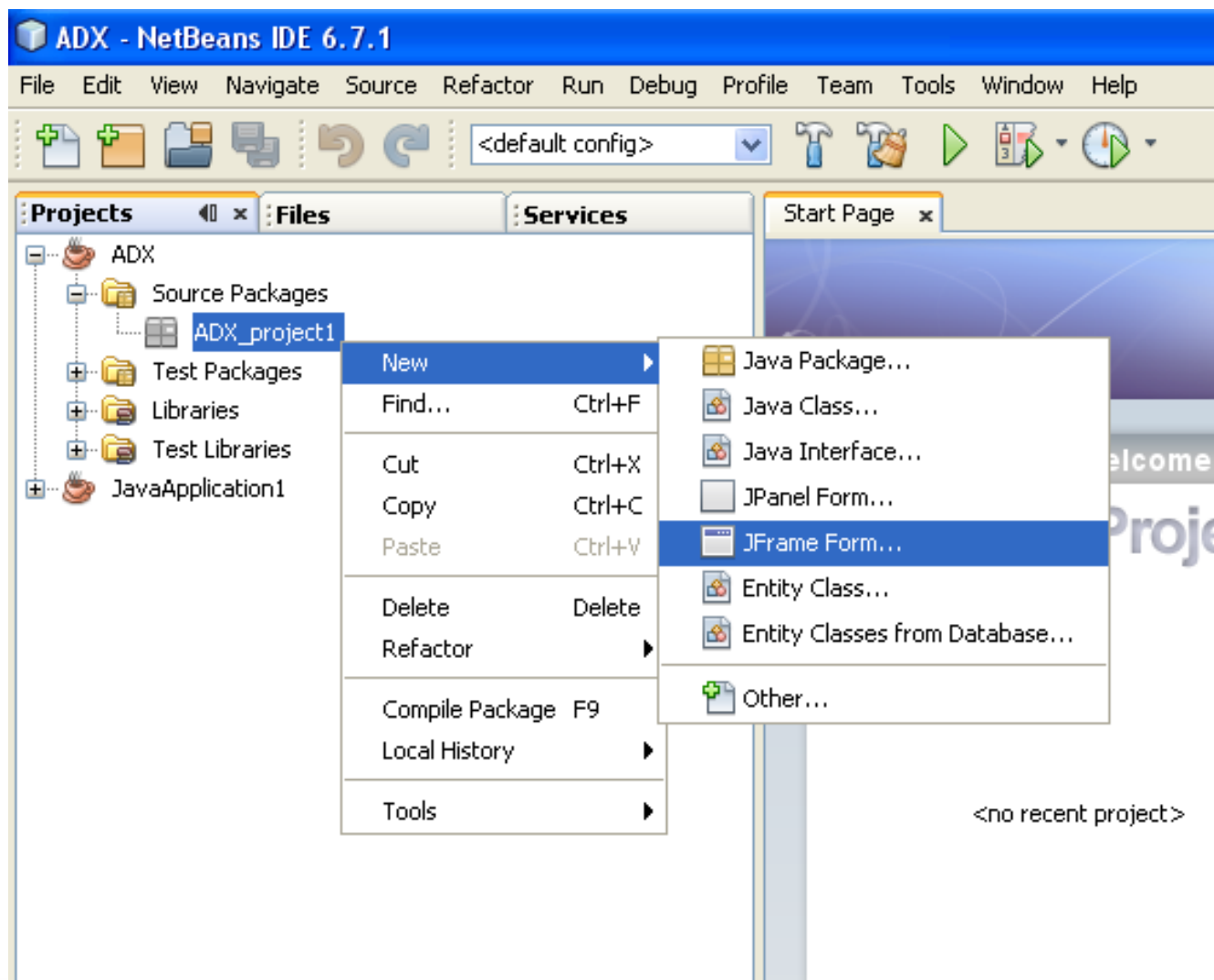
```
static invokeAndWait(Runnable obj)
```

Συνήθως προτιμούμε την πρώτη για desktop εφαρμογές ενώ τη δεύτερη για Web εφαρμογές.



**SWING ME NETBEANS**

# ΤΟ ΠΡΩΤΟ ΠΑΡΑΔΕΙΓΜΑ – ΜΕ GUI NETBEANS



New JFrame Form

Steps

1. Choose File Type

2. **Name and Location**

Name and Location

Class Name:

Project:

Location:

Package:

Created File:

< Back

Next >

Finish

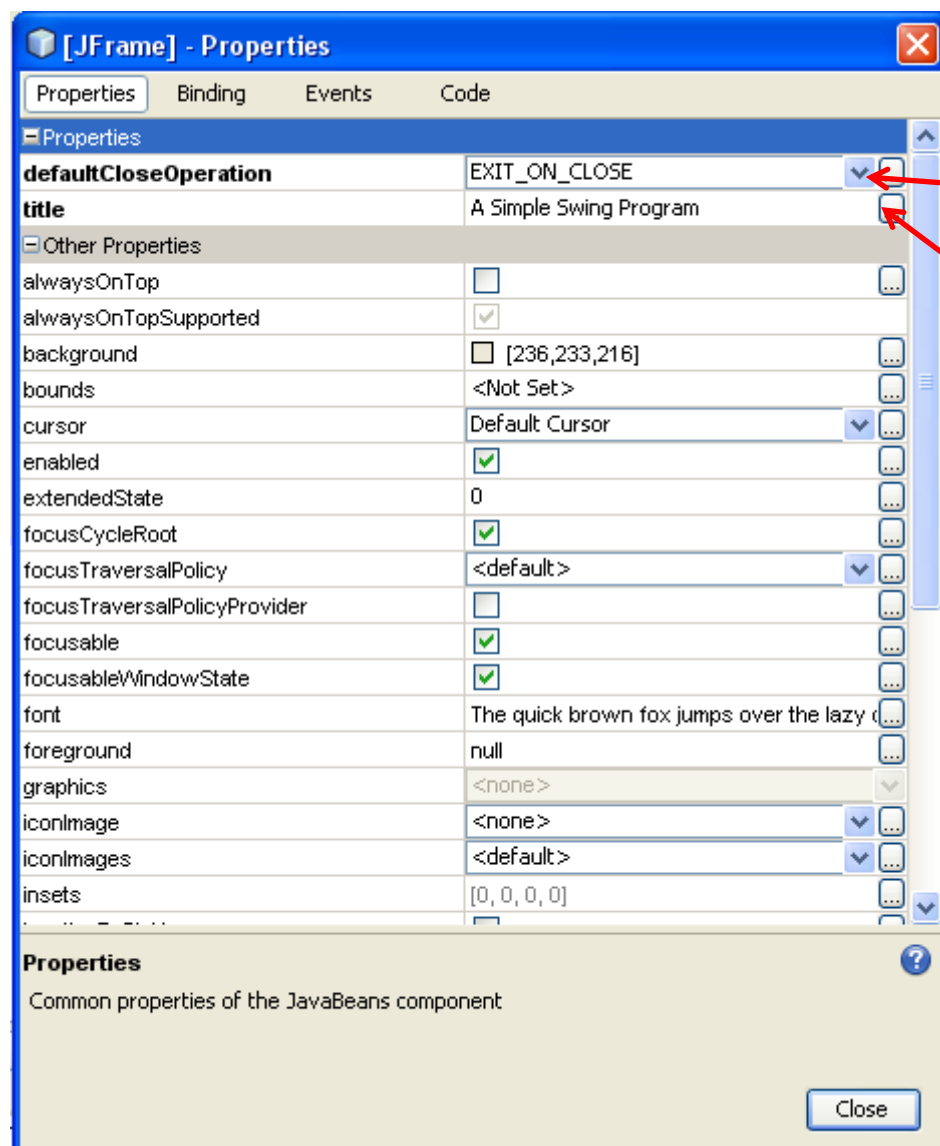
Cancel

Help

Δρ. Κεραμόπουλος Ευκλείδης

27

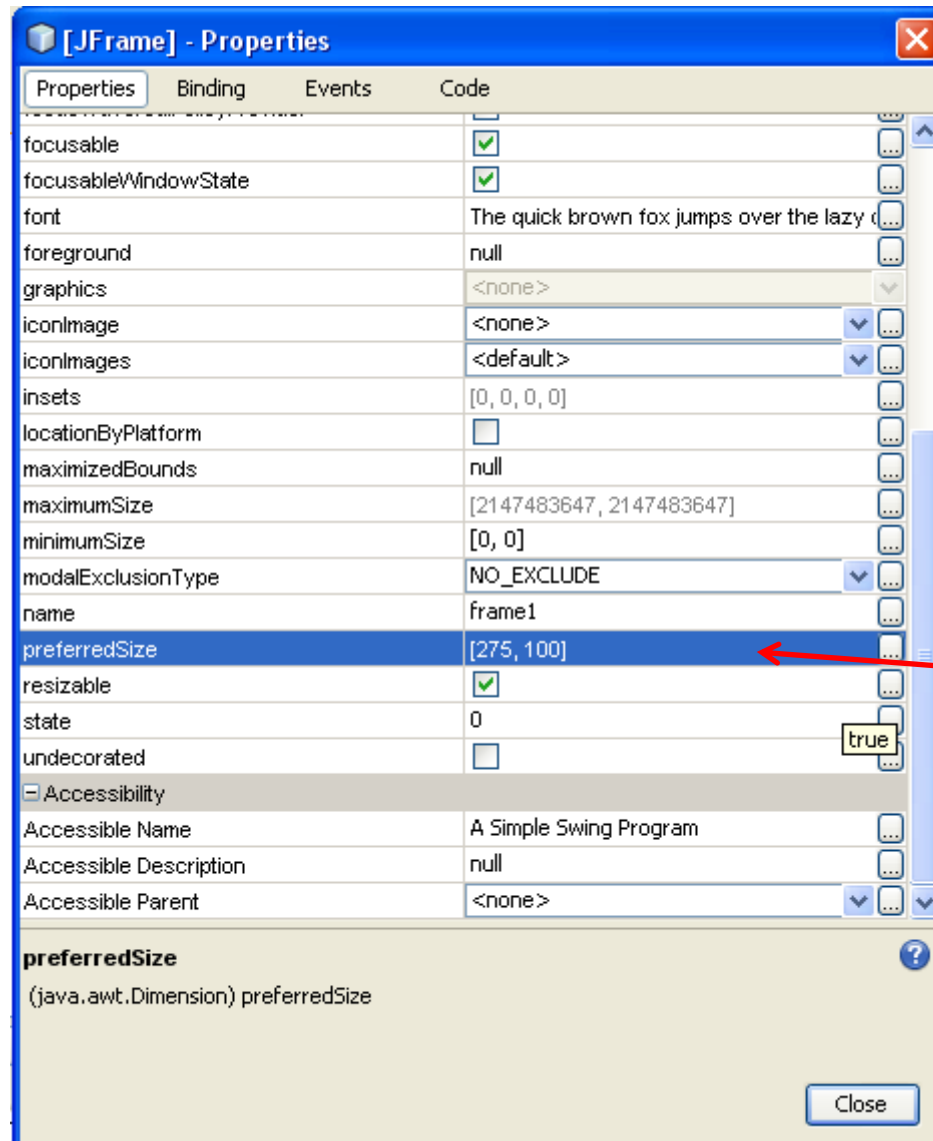
# ΔΕΞΙ ΠΛΗΚΤΡΟ ΠΑΝΩ ΣΤΟ FRAME ΚΑΙ ΕΠΙΛΕΓΩ PROPERTIES



Ορίζει τον τρόπο εξόδου

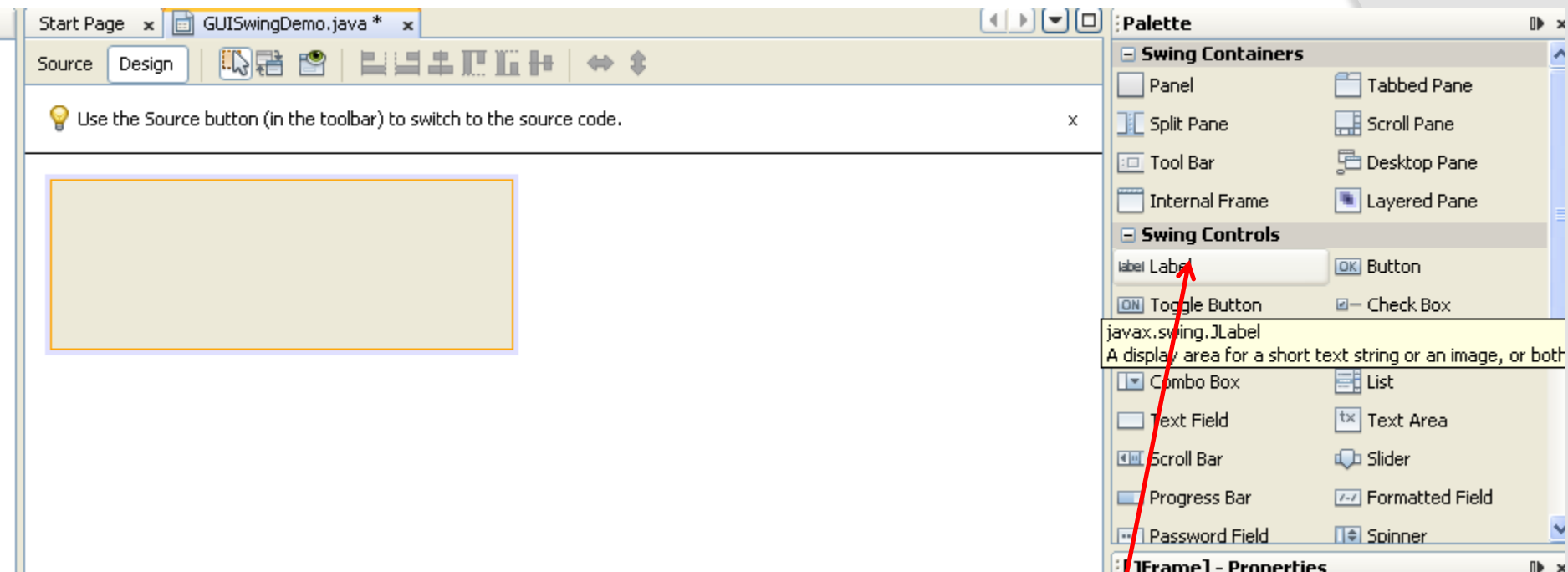
Ορίζει τον τίτλο του παράθυρου

# ΟΡΙΖΩ ΤΟ ΜΕΓΕΘΟΣ ΤΟΥ FRAME



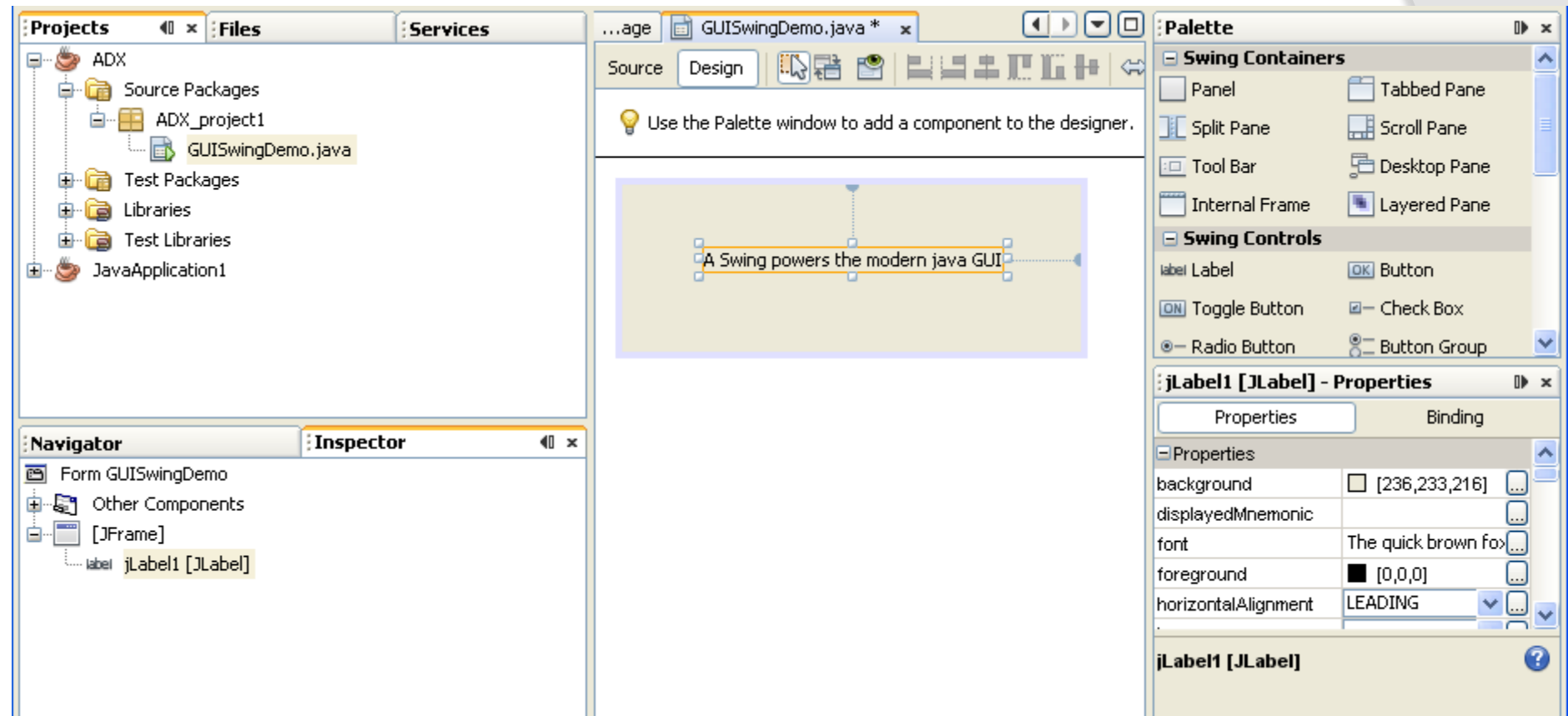
Ορίζει τον μέγεθος του  
παράθυρου

# ΕΠΙΛΕΓΩ ΤΟ LABEL

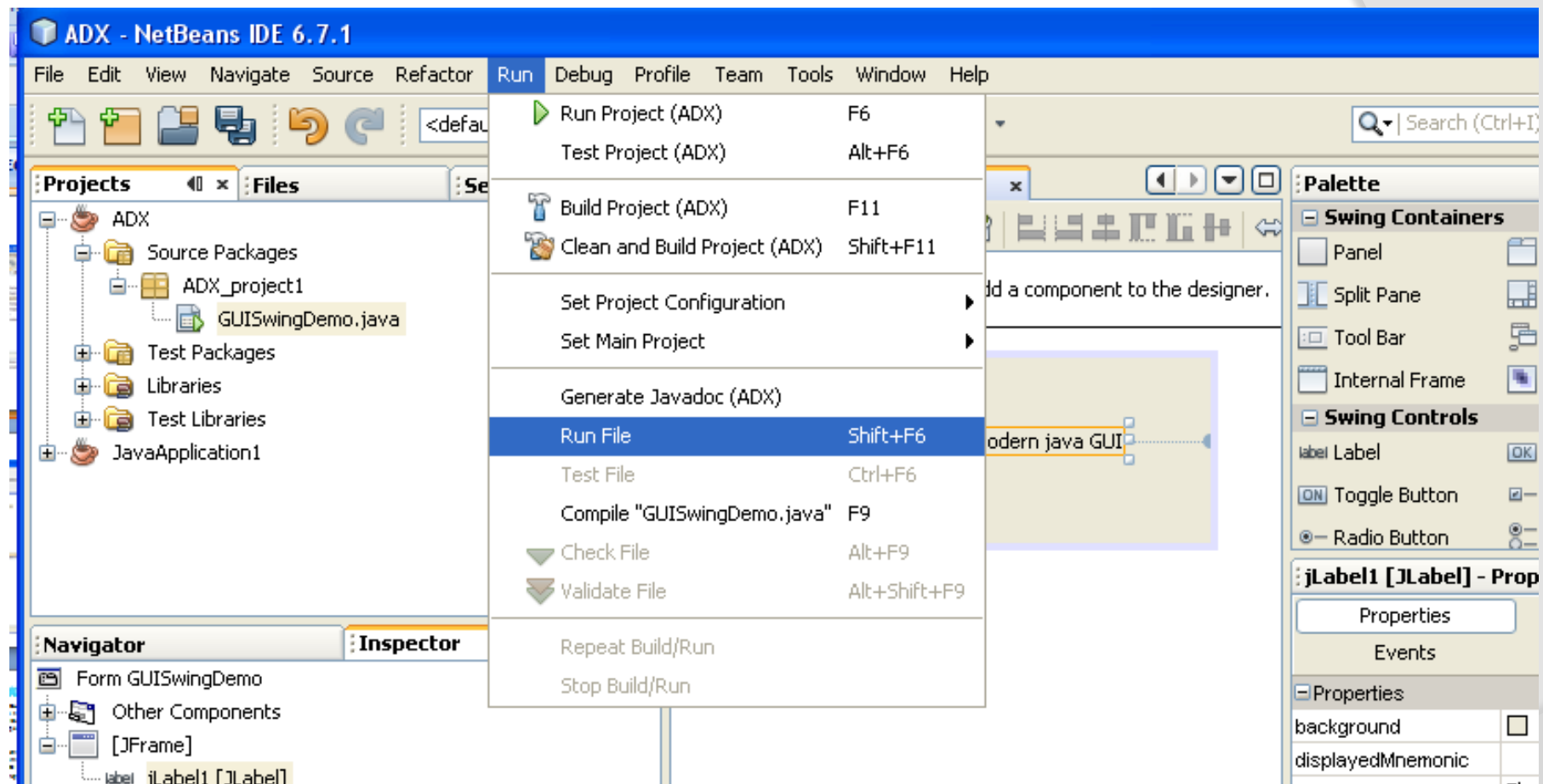


Επιλέγω το Label

# ΓΡΑΦΩ ΤΟ ΜΗΝΥΜΑ

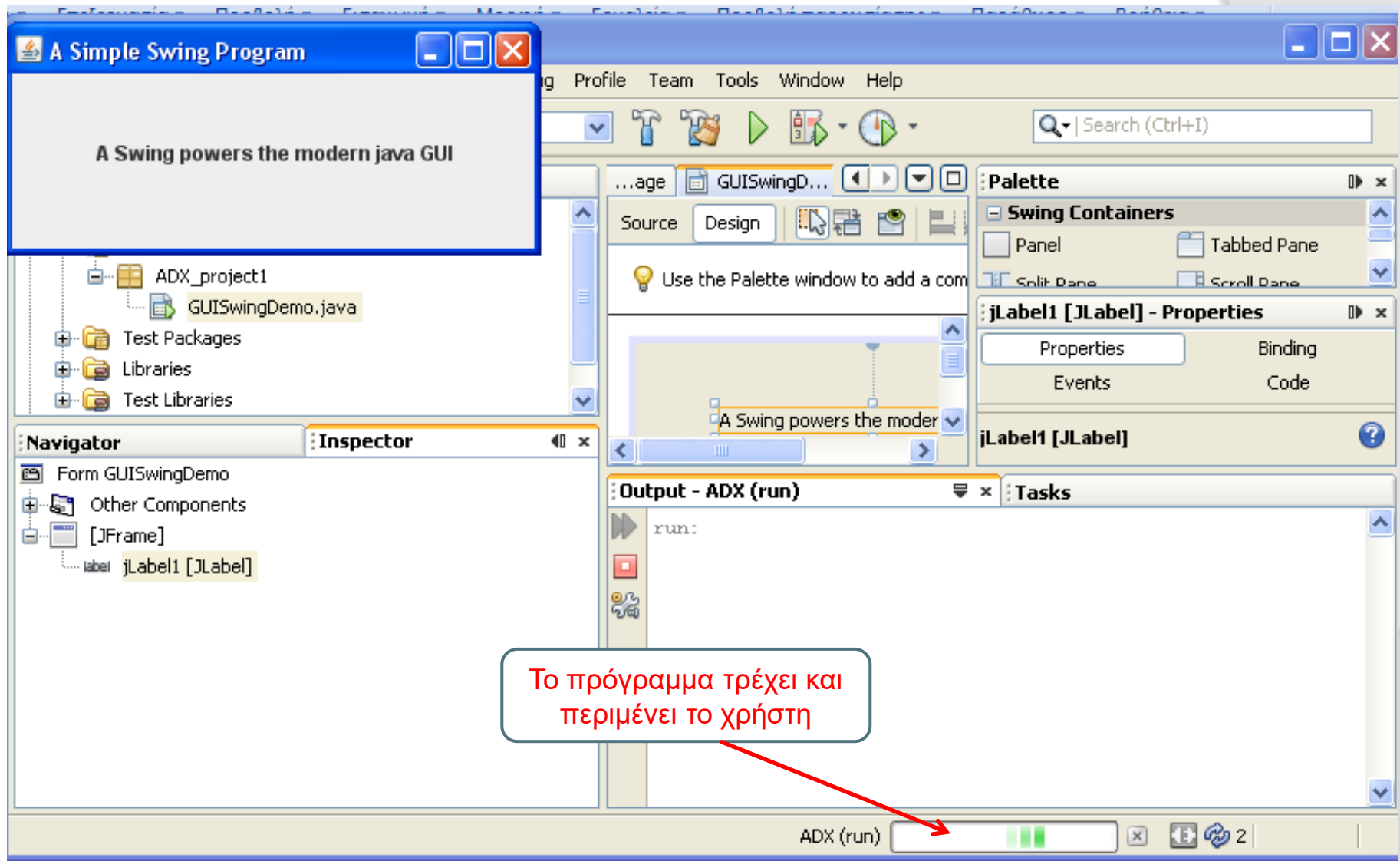


# ΕΚΤΕΛΩ ΤΟ ΠΡΟΓΡΑΜΜΑ





# ΕΚΤΕΛΕΣΗ ΤΟΥ ΠΡΟΓΡΑΜΜΑΤΟΣ



# Ο ΚΩΔΙΚΑΣ ΠΟΥ ΔΗΜΙΟΥΡΓΗΘΗΚΕ ΑΥΤΟΜΑΤΑ

```
1. package ADX_project1;

2. public class GUISwingDemo extends javax.swing.JFrame {

3. /** Creates new form GUISwingDemo */
4. public GUISwingDemo() {
5. initComponents();
6. }
7. /** This method is called from within the constructor to initialize the form.
8. * WARNING: Do NOT modify this code. The content of this method is always regenerated by the
9. Form Editor. */
9. @SuppressWarnings("unchecked")
10. // <editor-fold defaultstate="collapsed" desc="Generated Code">
11. private void initComponents() {
12. jLabel1 = new javax.swing.JLabel();
13. setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
14. setTitle("A Simple Swing Program");
15. jLabel1.setText("A Swing powers the modern java GUI");
16. javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
17. getContentPane().setLayout(layout);
```

# Ο ΚΩΔΙΚΑΣ ΠΟΥ ΔΗΜΙΟΥΡΓΗΘΗΚΕ ΑΥΤΟΜΑΤΑ

```
18. layout.setHorizontalGroup(
19. layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
20. .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup())
21. .addContainerGap(49, Short.MAX_VALUE)
22. .addComponent(jLabel1)
23. .addGap(47, 47, 47))
24.);
25. layout.setVerticalGroup(
26. layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
27. .addGroup(layout.createSequentialGroup())
28. .addGap(38, 38, 38)
29. .addComponent(jLabel1)
30. .addContainerGap(48, Short.MAX_VALUE))
31.);
32. pack();
33. }// </editor-fold>
```

# Ο ΚΩΔΙΚΑΣ ΠΟΥ ΔΗΜΙΟΥΡΓΗΘΗΚΕ ΑΥΤΟΜΑΤΑ

```
34. /**package A
35. * @param args the command line arguments
36. */
37. public static void main(String args[]) {
38. java.awt.EventQueue.invokeLater(new Runnable() {
39. public void run() {
40. new GUISwingDemo().setVisible(true);
41. }
42. });
43. }

44. // Variables declaration - do not modify
45. private javax.swing.JLabel jLabel1;
46. // End of variables declaration

47. }
```

# ΜΟΝΤΕΛΟ ΑΠΟΣΤΟΛΗΣ ΓΕΓΟΝΟΤΩΝ

- ◎ Πότε δημιουργείτε ένα γεγονός;
  - Mouse click
  - Keyboard click
  - Όταν περνάει κάποιο χρονικό διάστημα
- ◎ Πώς λειτουργεί το μοντέλο αποστολής γεγονότων (Event Handling);
  1. μόλις δημιουργείται ένα γεγονός (event) από ένα αντικείμενο που μπορεί να δημιουργήσει γεγονότα (event source), τότε
  2. δημιουργείται ένα αντικείμενο γεγονότος το οποίο περιέχει πληροφορίες για το γεγονός και για την πηγή από την οποία προέρχεται.
  3. Το γεγονός γίνεται αντιληπτό από τον ακροατή του γεγονότος (event listener) ο οποίος καλεί
  4. τον χειριστή του γεγονότος – μία μέθοδο που αντιστοιχεί σε κάποιο τύπο γεγονότος.

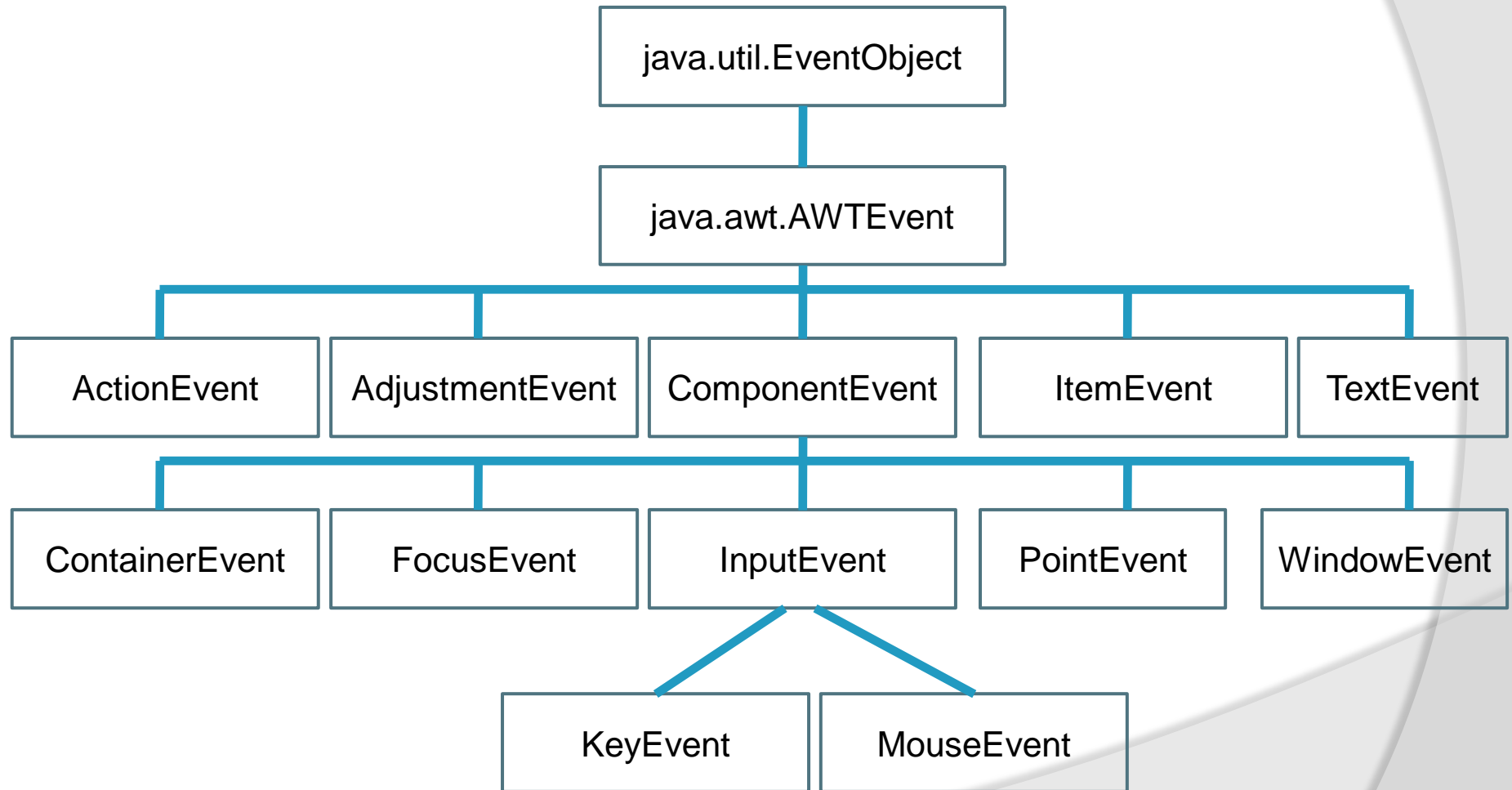
# ΜΟΝΤΕΛΟ ΑΠΟΣΤΟΛΗΣ ΓΕΓΟΝΟΤΩΝ

- ◎ Πώς γίνεται η σύνδεση μεταξύ της πηγής ενός γεγονότος και του ακροατή;
  - Κάθε πηγή γεγονότων παρέχει μεθόδους
    - για την καταχώρηση ακροατών χρησιμοποιούμε τη μέθοδο `addTypeListener( TypeListener el)`
    - για την αφαίρεση ακροατών χρησιμοποιούμε τη μέθοδο `removeTypeListener(TypeListener el)`
- ◎ Παράδειγμα
  - `jbtn.addKeyListener()` προσθέτει έναν ακροατή στο αντικείμενο `jbtn` για το πληκτρολόγιο
  - `jbtn.addMouseMotionListener()` προσθέτει έναν ακροατή στο αντικείμενο `jbtn` για την κίνηση του ποντικιού
  - `jbtn.removeKeyListener()` αφαιρεί από το αντικείμενο `jbtn` τον ακροατή για το πληκτρολόγιο
  - `jbtn.removeMouseMotionListener()` αφαιρεί από το αντικείμενο `jbtn` τον ακροατή για την κίνηση του ποντικιού

# ΑΚΡΟΑΤΕΣ

- Ο ακροατής είναι αντικείμενο το οποίο ενημερώνεται μόλις συμβεί ένα γεγονός.
- Για να τον χρησιμοποιήσουμε υπάρχουν δύο βασικές απαιτήσεις:
  1. Θα πρέπει να (π.χ. αν θέλουμε να προγραμματίσουμε το πάτημα ενός συγκεκριμένου κουμπιού) να προσθέσουμε έναν ακροατή για αυτό το γεγονός.
  2. Θα πρέπει να υλοποιηθεί η κατάλληλη μέθοδος η οποία θα δεχθεί το γεγονός και θα το υλοποιήσει.
- Έτοιμες μεθόδους για να δεχθούν και να επεξεργασθούν ένα γεγονός υπάρχουν στα παρακάτω interfaces:
  - java.awt.event
  - java.swing.event
  - java.beans
- Βασικός κανόνας για τη χρήση ενός ακροατή είναι η ταχύτητα. Αν χρειάζεται, θα πρέπει να υλοποιηθεί η επεξεργασία του ακροατή σε νέο νήμα.

# ΙΕΡΑΡΧΙΑ ΤΩΝ ΚΛΑΣΕΩΝ ΓΕΓΟΝΟΤΩΝ





# ΙΕΡΑΡΧΙΑ ΤΩΝ ΚΛΑΣΕΩΝ ΓΕΓΟΝΟΤΩΝ

- ⦿ Μία πηγή γεγονότων δεν δημιουργεί αποκλειστικά ένα γεγονός αλλά μία σειρά από γεγονότα.
  - Για παράδειγμα το πάτημα ενός κουμπιού δημιουργεί τα παρακάτω γεγονότα
    - `ActionEvent`
    - `ComponentEvent`
    - `FocusEvent`
    - `MouseEvent`
    - `KeyEvent`

# ΚΑΤΗΓΟΡΙΕΣ ΓΕΓΟΝΟΤΩΝ (JAVA.AWT.EVENT)

| Κατηγορία γεγονότος | Προέλευση                                                                                                                                                                     |
|---------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ActionEvent         | Δημιουργείται από κάποια ενέργεια του χρήστη, όπως πάτημα σε ένα κουμπί, διπλοπάτημα σε κάποιο από τα περιεχόμενα μιας λίστας ή ενός μενού, ή enter σε ένα στοιχείο κειμένου. |
| AdjustmentEvent     | Δημιουργείται από τη μεταβολή ενός ρυθμιζόμενου συστατικού, όπως η ράβδος κύλισης (scrollbar).                                                                                |
| ItemEvent           | Δημιουργείται όταν ο χρήστης επιλέγει ένα στοιχείο μιας λίστας ή πατάει σε ένα πλαίσιο ελέγχου ή σε ένα μενού επιλογών.                                                       |
| TextEvent           | Δημιουργείται όταν αλλάζει το περιεχόμενο ενός στοιχείου κειμένου (TextFiled, TextArea).                                                                                      |
| MouseEvent          | Δημιουργείται όταν ο χρήστης μετακινεί το ποντίκι ή επιλέγει ένα στοιχείο με αυτό.                                                                                            |
| KeyEvent            | Δημιουργείται όταν πιέζεται ή απελευθερώνεται ένα πλήκτρο του πληκτρολογίου.                                                                                                  |
| WindowEvent         | Δημιουργείται όταν συμβαίνει μετακίνηση, μεγιστοποίηση, ελαχιστοποίηση ή κλείσιμο του παραθύρου.                                                                              |
| FocusEvent          | Δημιουργείται όταν ένα συστατικό δέχεται ή χάνει την εστίαση (focus).                                                                                                         |
| ComponentEvent      | Δημιουργείται όταν ένα συστατικό αποκρύπτεται, εμφανίζεται, μετακινείται, ή αλλάζει μέγεθος.                                                                                  |
| ContainerEvent      | Δημιουργείται όταν ένα συστατικό προστίθεται ή αφαιρείται σε/από ένα συστατικό-υποδοχεά.                                                                                      |
| User-defined        | Οτιδήποτε άλλο                                                                                                                                                                |

# ΚΑΤΗΓΟΡΙΕΣ ΓΕΓΟΝΟΤΩΝ (JAVAX.SWING.EVENT)

| Κατηγορία γεγονότος | Προέλευση                                                                      |
|---------------------|--------------------------------------------------------------------------------|
| MouseEvent          | Δημιουργείται από την κίνηση της ρόδας του ποντικιού.                          |
| AncestorEvent       | Δημιουργείται από την προσθήκη, αφαίρεση ή μετακίνηση ενός συστατικού.         |
| CaretEvent          | Δημιουργείται όταν μετακινείται ο δείκτης του ποντικιού όταν γράφουμε κείμενο. |
| ChangeEvent         | Δημιουργείται όταν ένα συστατικό αλλάζει κατάσταση.                            |
| HyperlinkEvent      | Δημιουργείται όταν ένα hyperlink επιλέγεται.                                   |
| ListDataEvent       | Δημιουργείται όταν αλλάξουν τα περιεχόμενα μιας λίστας.                        |
| ListSelectionEvent  | Δημιουργείται όταν αλλάζει η επιλογή μας από μία λίστα.                        |
| MenuEvent           | Δημιουργείται όταν μία επιλογή μενού διαλέγεται.                               |
| TableModelEvent     | Δημιουργείται όταν το μοντέλο ενός πίνακα αλλάζει.                             |
| TreeExpansionEvent  | Δημιουργείται όταν ένα δέντρο «ανοίγει» ή «κλείνει».                           |
| TreeModelEvent      | Δημιουργείται όταν αλλάζει το μοντέλο του δέντρου.                             |
| TreeSelectionEvent  | Δημιουργείται όταν επιλέγεται ένα κλαδί του δέντρου.                           |

# ΧΡΗΣΙΜΕΣ ΜΕΘΟΔΟΙ ΓΕΓΟΝΟΤΩΝ

- ◉ **getSource():** ανήκει στην κλάση `java.util.EventObject` επιστρέφει σε μορφή `Object` την πηγή ενός γεγονότος.
- ◉ **getID():** ανήκει στην κλάση `java.awt.AWTEvent` επιστρέφει σε μορφή ακεραίου το ID του γεγονότος.

# ΤΟ ΔΕΥΤΕΡΟ ΠΑΡΑΔΕΙΓΜΑ

Δημιουργία μίας εφαρμογής όπου θα παρουσιάζεται ποιο κουμπί πατήθηκε.

# PUSH BUTTON

- Το push button είναι στιγμιότυπο ενός JButton το οποίο κληρονομεί την abstract class AbstractButton όπου ορίζεται η κοινή συμπεριφορά όλων των κουμπιών στην Java Swing.
- Ένα push button μπορεί να περιέχει κείμενο, εικόνα ή και τα δύο.
- Για να δημιουργήσουμε ένα push button με κείμενο χρησιμοποιούμε τον παρακάτω δομητή:
  - `JButton(String msg)`, όπου `msg` το κείμενο που θα περιέχει το κουμπί.
- Επίσης, για να προσθέσουμε ή να αφαιρέσουμε ακροατές για push buttons μας παρέχονται οι παρακάτω μέθοδοι:
  - `void addActionListener(ActionListener al);`
  - `void removeActionListener(ActionListener al);`
  - Το `al` ορίζει το αντικείμενο που θα δεχθεί μία ενημέρωση για το συγκεκριμένο γεγονός (πατήματος του συγκεκριμένου πλήκτρου)
- Μόλις ενημερωθεί το `al` για το γεγονός ενεργοποιεί τη μέθοδο:
  - `void actionPerformed(ActionEvent ae);`

# ΤΟ ΔΕΥΤΕΡΟ ΠΑΡΑΔΕΙΓΜΑ

```
1. import java.awt.*;
2. import java.awt.event.*;
3. import javax.swing.*;
4.
5. class ButtonDemo implements ActionListener {
6. JLabel jlab;
7.
8. ButtonDemo() {
9.
10.
11. JFrame jfrm = new JFrame("A Button Example");
12.
13. jfrm.setLayout(new FlowLayout());
14.
15. jfrm.setSize(220, 90);
16.
17. jfrm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

Περιέχει την FlowLayout τάξη

Περιέχει την ActionListener και  
ActionEvent τάξη

Η τάξη ButtonDemo μπορεί να  
χειριστεί γεγονότα επειδή  
Implements ActionListener

Το frame έχει διάταξη  
FlowLayout, δηλαδή σε μία σειρά.

# ΤΟ ΔΕΥΤΕΡΟ ΠΑΡΑΔΕΙΓΜΑ

```
18. JButton jbtnFirst = new JButton("First");
19. JButton jbtnSecond = new JButton("Second");
20.
21. jbtnFirst.addActionListener(this);
22. jbtnSecond.addActionListener(this);
23.
24. jfrm.add(jbtnFirst);
25. jfrm.add(jbtnSecond);
26.
27. jlab = new JLabel("Press a button.");
28.
29. jfrm.add(jlab);
30.
31. jfrm.setVisible(true);
32. }
```

← Δημιουργείται ένα κουμπί που περιέχει το κείμενο «First»

← Ορίζουμε ακροατές στα κουμπιά το γεγονός Action

← Προσθέτουμε τα κουμπιά στο Frame

← Προσθέτουμε την ετικέτα στο Frame



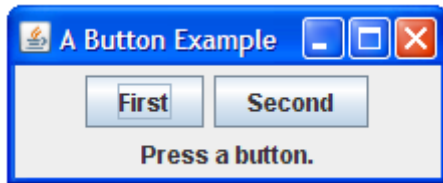
# ΤΟ ΔΕΥΤΕΡΟ ΠΑΡΑΔΕΙΓΜΑ

```
33. public void actionPerformed(ActionEvent ae) {
34. if(ae.getActionCommand().equals("First"))
35. jlab.setText("First button was pressed.");
36. else
37. jlab.setText("Second button was pressed. ");
38. }
39.
40. public static void main(String args[]) {
41. SwingUtilities.invokeLater(new Runnable() {
42. public void run() {
43. new ButtonDemo();
44. }
45. });
46.
47. }
48. }
```

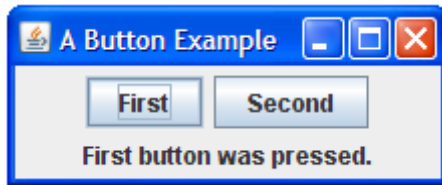


Όταν συμβαίνει ένα γεγονός τότε αυτό διαχειρίζεται από τη μέθοδο `actionPerformed`. Το αντικείμενο `ActionEvent` αντιπροσωπεύει το γεγονός που δημιουργείται από το κουμπί και περνάει ως αντικείμενο. Με τη μέθοδο `getActionCommand` μαθαίνουμε ποιο κουμπί πατήθηκε. Το όνομα του κουμπιού είναι το κείμενο που περιέχει.

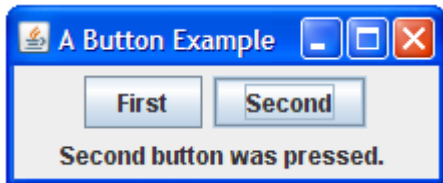
# ΕΚΤΕΛΕΣΗ



Εκτέλεση



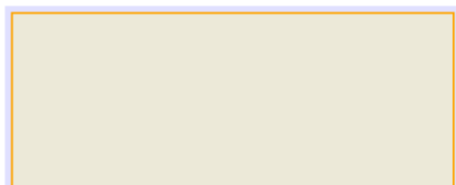
Πάτημα του  
κουμπιού «First»



Πάτημα του  
κουμπιού «Second»

# ΤΟ ΔΕΥΤΕΡΟ ΠΑΡΑΔΕΙΓΜΑ – GUI NETBEANS

The Preview Design button (in the toolbar) enables the Preview Design button.



Δημιουργήσαμε ένα  
νέο αρχείο java ως  
JFrame Form.  
Ορίσαμε τις ιδιότητες  
title

[JFrame] - Properties

Properties Binding Events Code

Properties

|                       |                  |
|-----------------------|------------------|
| defaultCloseOperation | EXIT_ON_CLOSE    |
| title                 | A Button Example |

Other Properties

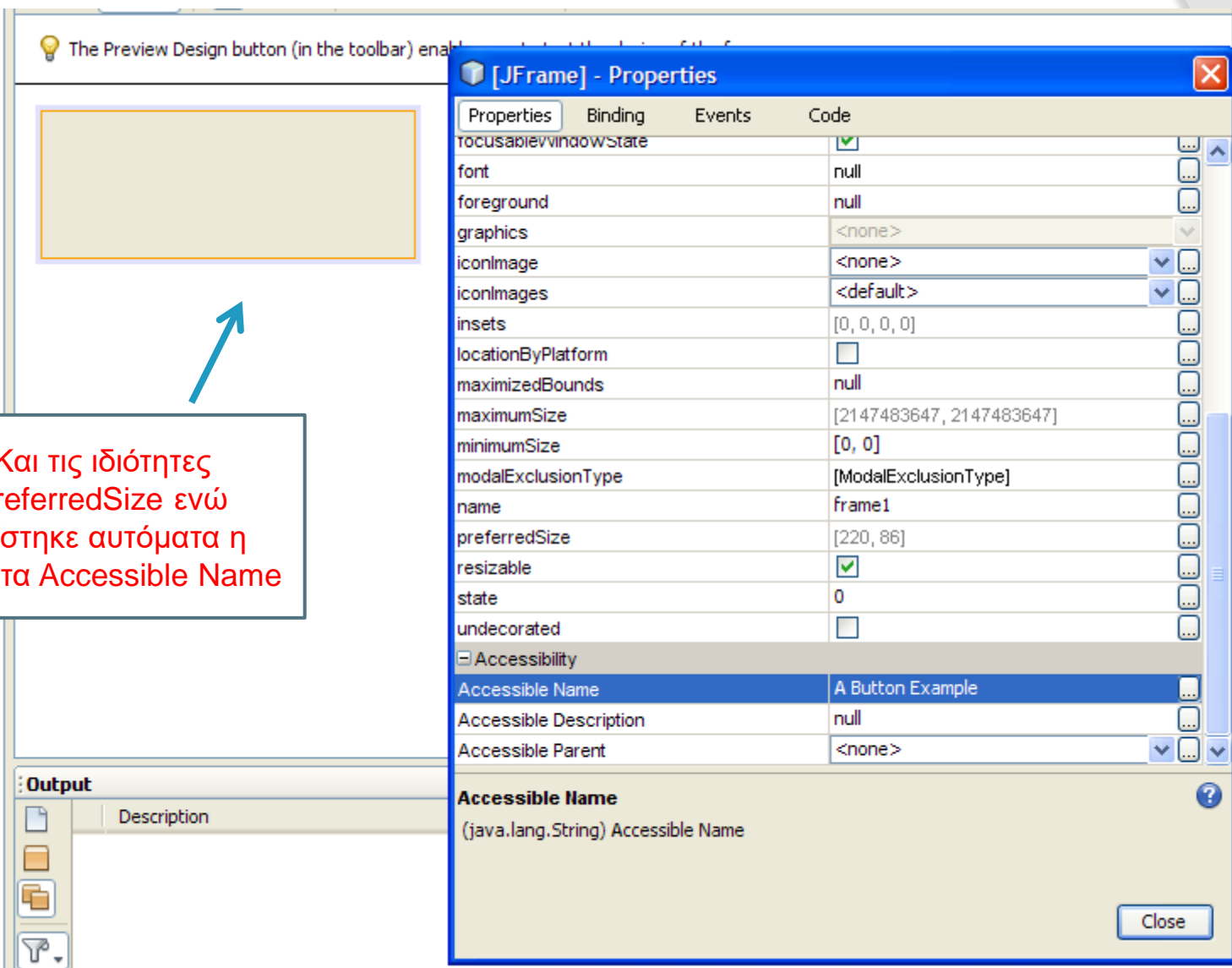
|                              |                                     |
|------------------------------|-------------------------------------|
| alwaysOnTop                  | <input type="checkbox"/>            |
| alwaysOnTopSupported         | <input checked="" type="checkbox"/> |
| background                   | [236,233,216]                       |
| bounds                       | <Not Set>                           |
| cursor                       | Default Cursor                      |
| enabled                      | <input checked="" type="checkbox"/> |
| extendedState                | 0                                   |
| focusCycleRoot               | <input checked="" type="checkbox"/> |
| focusTraversalPolicy         | <default>                           |
| focusTraversalPolicyProvider | <input type="checkbox"/>            |
| focusable                    | <input checked="" type="checkbox"/> |
| focusableWindowState         | <input checked="" type="checkbox"/> |
| font                         | null                                |
| foreground                   | null                                |
| graphics                     | <none>                              |
| iconImage                    | <none>                              |
| iconImages                   | <default>                           |

Accessible Name  
(java.lang.String) Accessible Name

Close

# ΤΟ ΔΕΥΤΕΡΟ ΠΑΡΑΔΕΙΓΜΑ – GUI NETBEANS

The Preview Design button (in the toolbar) enables the Preview Design mode.



Kαι τις ιδιότητες preferredSize ενώ ορίστηκε αυτόματα η ιδιότητα Accessible Name

| Properties             | Binding | Events                              | Code                     |
|------------------------|---------|-------------------------------------|--------------------------|
| focusableWindowState   |         | <input checked="" type="checkbox"/> | ...                      |
| font                   |         |                                     | null                     |
| foreground             |         |                                     | null                     |
| graphics               |         |                                     | <none>                   |
| iconImage              |         |                                     | <none>                   |
| iconImages             |         |                                     | <default>                |
| insets                 |         |                                     | [0, 0, 0, 0]             |
| locationByPlatform     |         | <input type="checkbox"/>            | ...                      |
| maximizedBounds        |         |                                     | null                     |
| maximumSize            |         |                                     | [2147483647, 2147483647] |
| minimumSize            |         |                                     | [0, 0]                   |
| modalExclusionType     |         |                                     | [ModalExclusionType]     |
| name                   |         |                                     | frame1                   |
| preferredSize          |         |                                     | [220, 86]                |
| resizable              |         | <input checked="" type="checkbox"/> | ...                      |
| state                  |         |                                     | 0                        |
| undecorated            |         | <input type="checkbox"/>            | ...                      |
| Accessibility          |         |                                     |                          |
| Accessible Name        |         |                                     | A Button Example         |
| Accessible Description |         |                                     | null                     |
| Accessible Parent      |         |                                     | <none>                   |

**Accessible Name**  
(java.lang.String) Accessible Name

Close

# ΤΟ ΔΕΥΤΕΡΟ ΠΑΡΑΔΕΙΓΜΑ – GUI NETBEANS

Προσθέσαμε ένα Button και ορίσαμε την ιδιότητα του text

The screenshot shows the NetBeans IDE with the 'Design' tab selected. A button labeled 'First Button' is visible in the design view. The 'Properties' window is open, showing the 'Properties' tab for 'jButton1 [JButton]'. The 'text' property is highlighted in the 'Properties' list. The 'Other Properties' list is also visible below.

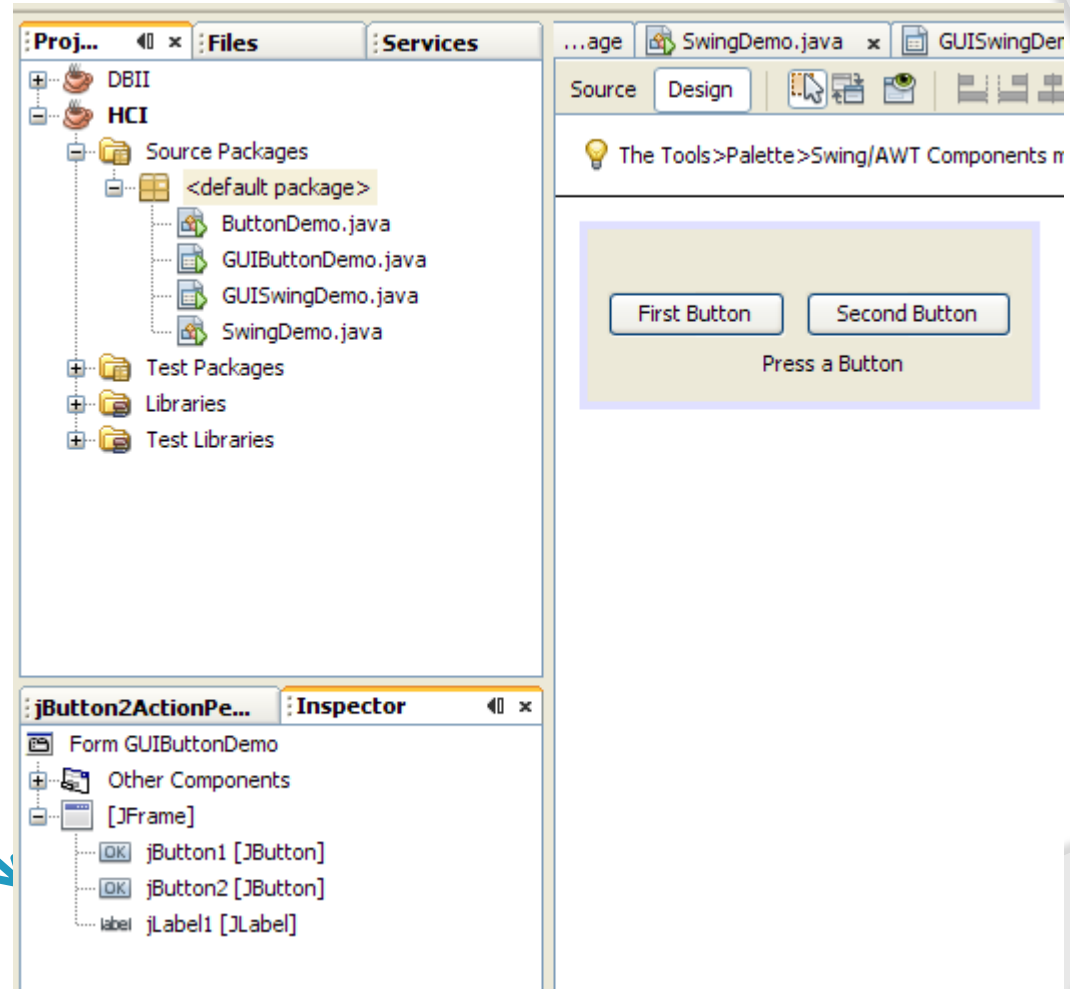
| Properties  |                 |
|-------------|-----------------|
| action      |                 |
| background  | [236,233,216]   |
| font        | Tahoma 11 Plain |
| foreground  | [0,0,0]         |
| icon        |                 |
| mnemonic    |                 |
| <b>text</b> | First Button    |
| toolTipText | null            |

| Other Properties       |                                     |
|------------------------|-------------------------------------|
| UIClassID              | ButtonUI                            |
| actionCommand          | First Button                        |
| alignmentX             | 0.0                                 |
| alignmentY             | 0.5                                 |
| autoscrolls            |                                     |
| baselineResizeBehavior | [BaselineResizeBehavior]            |
| border                 | [XPEmptyBorder]                     |
| borderPainted          | <input checked="" type="checkbox"/> |
| buttonGroup            | <none>                              |
| componentPopupMenu     | <none>                              |
| contentAreaFilled      | <input checked="" type="checkbox"/> |
| debugGraphicsOptions   | NO_CHANGES                          |
| defaultColor           |                                     |

**Other Properties**  
Other properties of the JavaBeans component

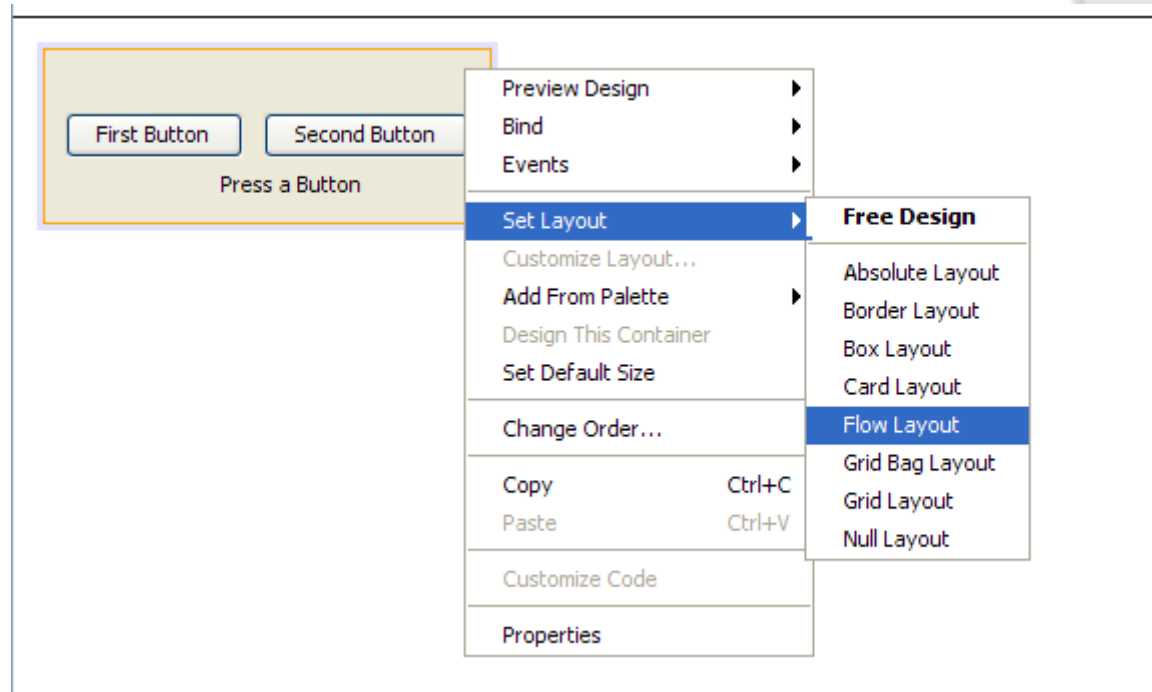
# ΤΟ ΔΕΥΤΕΡΟ ΠΑΡΑΔΕΙΓΜΑ – GUI NETBEANS

Προσθέσαμε ακόμα ένα Button με text «Second Button» και ένα Label με text «Press a Button»

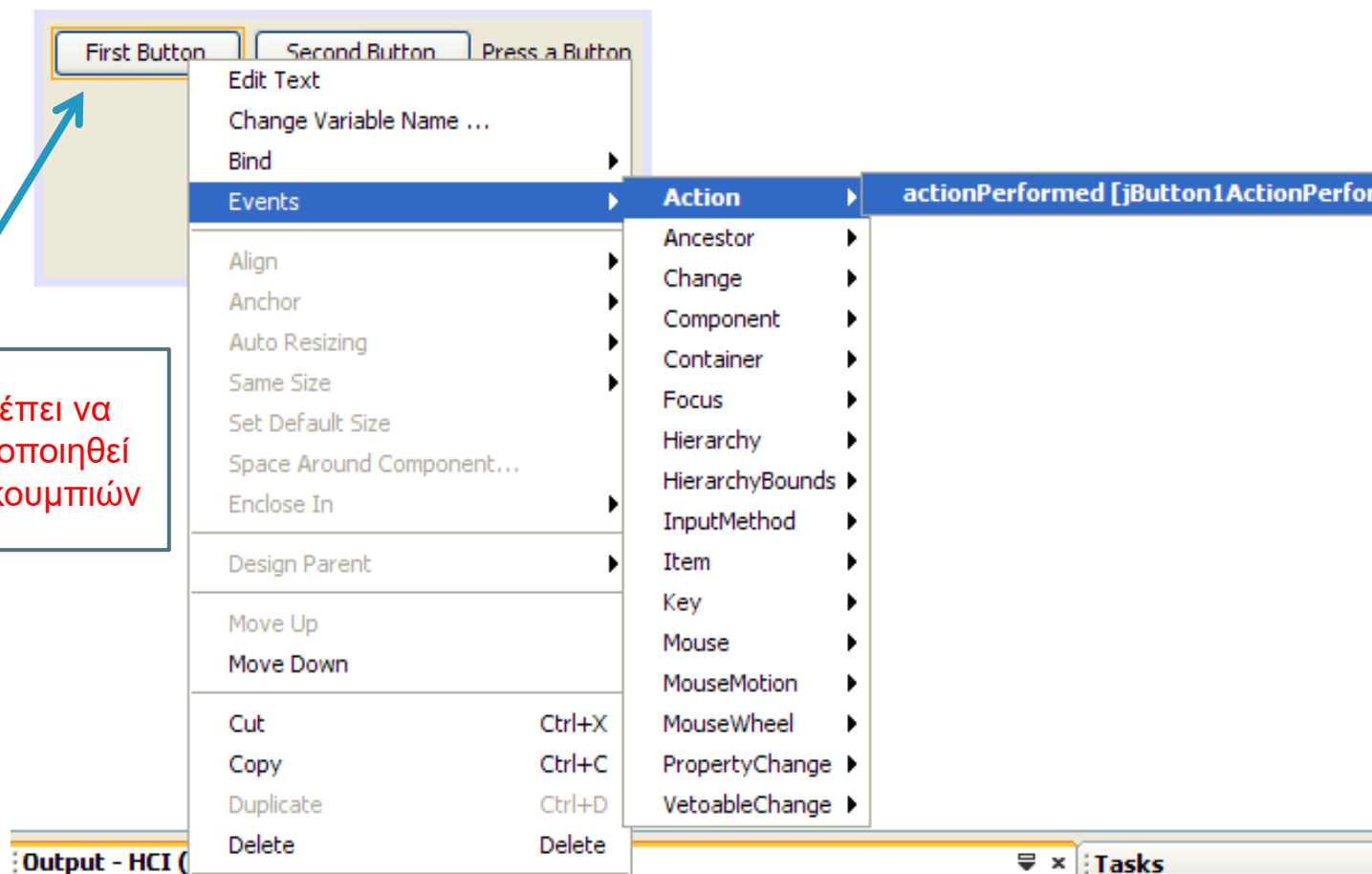


# ΤΟ ΔΕΥΤΕΡΟ ΠΑΡΑΔΕΙΓΜΑ – GUI NETBEANS

Ορίζουμε η διάταξη του  
Frame να είναι  
FlowLayout



# ΤΟ ΔΕΥΤΕΡΟ ΠΑΡΑΔΕΙΓΜΑ – GUI NETBEANS





# ΤΟ ΔΕΥΤΕΡΟ ΠΑΡΑΔΕΙΓΜΑ – GUI NETBEANS

The screenshot displays the NetBeans IDE interface. On the left, the 'main - Navigator' window shows the 'Members View' of the 'GUIButtonDemo' project. The project structure includes a package 'GUIButtonDemo' containing a class 'JFrame'. The class has methods 'initComponents()', 'jButton1ActionPerformed(ActionEvent evt)', 'jButton2ActionPerformed(ActionEvent evt)', and 'main(String[] args)'. The 'main' method is highlighted. Below the package, the components are listed: 'jButton1 : JButton', 'jButton2 : JButton', and 'jLabel1 : JLabel'. A blue arrow points from the 'main' method in the Members View to the 'main' method in the source code editor.

The source code editor shows the following code:

```
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
 // TODO add your handling code here:
 jLabel1.setText("Second Button was pressed");
}

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
 // TODO add your handling code here:
 jLabel1.setText("First Button was pressed");
}

/**
 * @param args the command line arguments
 */
public static void main(String args[]) {
 java.awt.EventQueue.invokeLater(new Runnable() {
 public void run() {
 initComponents();
 jButton1ActionPerformed(null);
 jButton2ActionPerformed(null);
 }
 });
}
```

The 'Output - HCI (run)' window at the bottom shows the output of the program:

```
run:
BUILD SUCCESSFUL (total time: 9 seconds)
```

A blue arrow points from the 'Output - HCI (run)' window to the 'main' method in the source code editor.

Μπορούμε να δούμε όλα  
τα αντικείμενα της  
εφαρμογής

Προγραμματίζουμε τι θα  
συμβεί όταν πατηθεί το  
κάθε κουμπί

# ΤΟ ΤΡΙΤΟ ΠΑΡΑΔΕΙΓΜΑ

Δημιουργία μίας εφαρμογής υλοποίησης χρονομέτρου. Θα περιέχει δύο κουμπιά, ένα start και ένα stop. Θα εμφανίζει το χρόνο που μεσολαβεί ανάμεσα στο πάτημα των δύο πλήκτρων.

# ΤΟ ΤΡΙΤΟ ΠΑΡΑΔΕΙΓΜΑ

```
1. import java.awt.*;
2. import java.awt.event.*;
3. import javax.swing.*;
4. import java.util.*;
5.
6. class Stopwatch implements ActionListener {
7. JLabel jlab;
8. long start;
9. Stopwatch() {
10. JFrame jfrm = new JFrame("A Simple Stopwatch");
11. jfrm.setLayout(new FlowLayout());
12. jfrm.setSize(230, 90);
13. jfrm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
14. JButton jbtnStart = new JButton("Start");
15. JButton jbtnStop = new JButton("Stop");
```

Περιέχει την Calendar τάξη



# ΤΟ ΤΡΙΤΟ ΠΑΡΑΔΕΙΓΜΑ

```
16. jbtnStart.addActionListener(this);
17. jbtnStop.addActionListener(this);
18.
19. jfrm.add(jbtnStart);
20. jfrm.add(jbtnStop);
21.
22. jlab = new JLabel("Press Start to begin timing.");
23. jfrm.add(jlab);
24. jfrm.setVisible(true);
25. }
```

# ΤΟ ΤΡΙΤΟ ΠΑΡΑΔΕΙΓΜΑ

```
26. public void actionPerformed(ActionEvent ae) {
27. Calendar cal = Calendar.getInstance();
28. if(ae.getActionCommand().equals("Start")) {
29. start = cal.getTimeInMillis();
30. jlab.setText("Stopwatch is Running...");
31. }
32. else
33. jlab.setText("Elapsed time is "
34. + (double) (cal.getTimeInMillis() - start)/1000);
35. }
36.
37. public static void main(String args[]) {
38. SwingUtilities.invokeLater(new Runnable() {
39. public void run() {
40. new StopWatch();
41. }
42. });
43. }
44. }
```

Παίρνουμε την ώρα του συστήματος.

Παίρνουμε την ώρα του συστήματος σε milliseconds

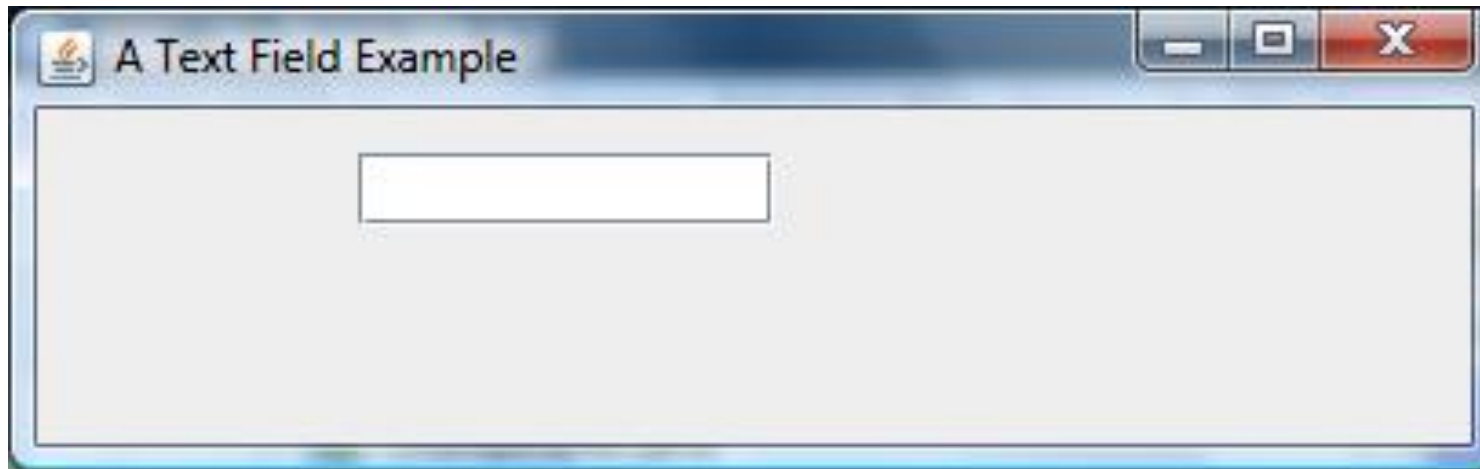
Η διαφορά μετατρέπεται σε δευτερόλεπτα

# ΧΡΗΣΗ ΚΕΙΜΕΝΟΥ

- ⦿ JTextField: Μπορεί ο χρήστης να εισάγει μία γραμμή κειμένου.
- ⦿ Πατώντας το πλήκτρο <ENTER> δημιουργείται ένα γεγονός το οποίο μπορεί να καλυφθεί προσθέτοντας ActionListener και προγραμματίζοντας το actionPerformed του JTextField
- ⦿ Χρήσιμες μέθοδοι:
  - getText(): παίρνουμε το κείμενο που έχουμε εισάγει σε ένα jTextField
  - setText(): γράφουμε κείμενο σε ένα jTextField
  - setActionCommand(String): ορίζουμε σε ποιο όνομα θα αντιδρά ένα jTextField

# ΤΟ ΤΕΤΑΡΤΟ ΠΑΡΑΔΕΙΓΜΑ

Δημιουργία μίας εφαρμογής για να γίνει επίδειξη χρήσης κειμένου στην Java Swing.



# ΤΟ ΤΕΤΑΡΤΟ ΠΑΡΑΔΕΙΓΜΑ

```
1. import java.awt.*;
2. import java.awt.event.*;
3. import javax.swing.*;
4.
5. class JTextFieldDemo implements ActionListener {
6.
7. JTextField jtf;
8. JLabel jlab;
9.
10. JTextFieldDemo() {
11. JFrame jfrm = new JFrame("A Text Field Example");
12. jfrm.setLayout(new FlowLayout());
13. jfrm.setSize(240, 90);
14. jfrm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

← Δημιουργία αντικειμένου TextField



# ΤΟ ΤΕΤΑΡΤΟ ΠΑΡΑΔΕΙΓΜΑ

```
15. jtf = new JTextField(10);
16. jtf.addActionListener(this);
17. jfrm.add(jtf);
18. jlab = new JLabel("");
19. jfrm.add(jlab);
20. jfrm.setVisible(true);
21. }
22.
23. public void actionPerformed(ActionEvent ae) {
24. jlab.setText("Current contents: " + jtf.getText());
25. }
26.
27. public static void main(String args[]) {
28. SwingUtilities.invokeLater(new Runnable() {
29. public void run() {
30. new JTextFieldDemo();
31. }
32. });
33. }
34. }
```

# ΤΟ ΠΕΜΠΤΟ ΠΑΡΑΔΕΙΓΜΑ

Δημιουργία μίας εφαρμογής για να γίνει επίδειξη χρήσης δύο περιοχών κειμένου στην Java Swing.



# ΤΟ ΠΕΜΠΤΟ ΠΑΡΑΔΕΙΓΜΑ

```
1. import java.awt.*;
2. import java.awt.event.*;
3. import javax.swing.*;
4.
5. class TwoTFDemo implements ActionListener {
6.
7. JTextField jtf1;
8. JTextField jtf2;
9. JLabel jlab;
10.
11. TwoTFDemo() {
12. JFrame jfrm = new JFrame("Use Two Text Fields");
13. jfrm.setLayout(new FlowLayout());
14. jfrm.setSize(240, 120);
15. jfrm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
16. jtf1 = new JTextField(10);
17. jtf2 = new JTextField(10);
```

# ΤΟ ΠΕΜΠΤΟ ΠΑΡΑΔΕΙΓΜΑ

```
18. jtf1.setActionCommand("One");
19. jtf2.setActionCommand("Two");
20.
21. jtf1.addActionListener(this);
22. jtf2.addActionListener(this);
23. jfrm.add(jtf1);
24. jfrm.add(jtf2);
25. jlab = new JLabel("");
26. jfrm.add(jlab);
27. jfrm.setVisible(true);
28. }
29.
30. public void actionPerformed(ActionEvent ae) {
31. if(ae.getActionCommand().equals("One"))
32. jlab.setText("ENTER pressed in tf1: " + jtf1.getText());
33. else
34. jlab.setText("ENTER pressed in jtf2: " + jtf2.getText());
35. }
```



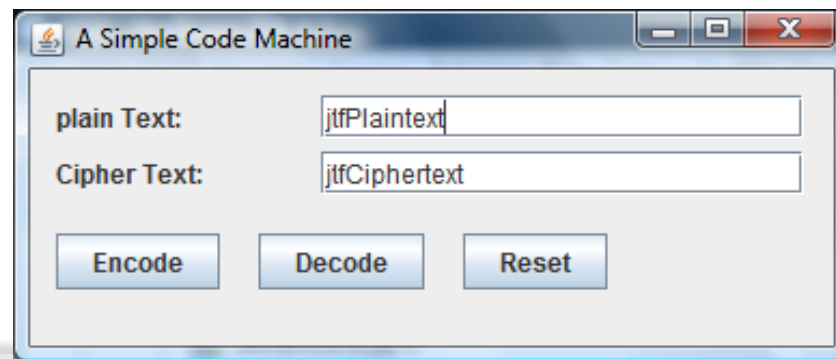
Ορίζουμε τα ονόματα των  
TextField

# ΤΟ ΠΕΜΠΤΟ ΠΑΡΑΔΕΙΓΜΑ

```
36. public static void main(String args[]) {
37. SwingUtilities.invokeLater(new Runnable() {
38. public void run() {
39. new TwoTFDDemo();
40. }
41. });
42. }
43. }
```

# ΤΟ ΕΚΤΟ ΠΑΡΑΔΕΙΓΜΑ

Δημιουργία μίας εφαρμογής υλοποίησης ενός απλού κωδικοποιητή-αποκωδικοποιητή. Η εφαρμογή αποτελείται από τρία κουμπιά (Code, Decode, Reset), δύο περιοχές κειμένου (μία για το code και μία για το decode) και δύο ετικέτες για το χαρακτηρισμό των περιοχών κειμένου.



# ΤΟ ΕΚΤΟ ΠΑΡΑΔΕΙΓΜΑ

```
1. import java.awt.*;
2. import java.awt.event.*;
3. import javax.swing.*;
4.
5. class Coder implements ActionListener {
6.
7. JTextField jtfPlaintext;
8. JTextField jtfCiphertext;
9.
10. Coder() {
11. JFrame jfrm = new JFrame("A Simple Code Machine");
12. jfrm.setLayout(new FlowLayout());
13. jfrm.setSize(340, 120);
14. jfrm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
15. JLabel jlabPlaintext = new JLabel(" Plain Text: ");
16. JLabel jlabCiphertext = new JLabel("Cipher Text: ");
17. jtfPlaintext = new JTextField(20);
18. jtfCiphertext = new JTextField(20);
```

# ΤΟ ΕΚΤΟ ΠΑΡΑΔΕΙΓΜΑ

```
19. jtfPlaintext.setActionCommand("Encode");
20. jtfCiphertext.setActionCommand("Decode");
21.
22. jtfPlaintext.addActionListener(this);
23. jtfCiphertext.addActionListener(this);
24.
25. jfrm.add(jlabPlaintext);
26. jfrm.add(jtfPlaintext);
27. jfrm.add(jlabCiphertext);
28. jfrm.add(jtfCiphertext);
29.
30. JButton jbtnEncode = new JButton("Encode");
31. JButton jbtnDecode = new JButton("Decode");
32. JButton jbtnReset = new JButton("Reset");
33.
34. jbtnEncode.addActionListener(this);
35. jbtnDecode.addActionListener(this);
36. jbtnReset.addActionListener(this);
```



# ΤΟ ΕΚΤΟ ΠΑΡΑΔΕΙΓΜΑ

```
37. jfrm.add(jbtnEncode);
38. jfrm.add(jbtnDecode);
39. jfrm.add(jbtnReset);
40. jfrm.setVisible(true);
41. }
42.
43. public void actionPerformed(ActionEvent ae) {
44. if(ae.getActionCommand().equals("Encode")) {
45. StringBuilder str = new StringBuilder(jtfPlaintext.getText());
46.
47. for(int i=0; i<str.length(); i++)
48. str.setCharAt(i, (char)(str.charAt(i) + 1));
49.
50. jtfCiphertext.setText(str.toString());
51. }
52.
53. }
```

# ΤΟ ΕΚΤΟ ΠΑΡΑΔΕΙΓΜΑ

```
54. else if(ae.getActionCommand().equals("Decode")) {
55. StringBuilder str = new StringBuilder(jtfCiphertext.getText());
56. for(int i=0; i<str.length(); i++)
57. str.setCharAt(i, (char)(str.charAt(i) - 1));
58. jtfPlaintext.setText(str.toString());
59. }
60. else {
61. jtfPlaintext.setText("");
62. jtfCiphertext.setText("");
63. }
64. }
65.
66. public static void main(String args[]) {
67. SwingUtilities.invokeLater(new Runnable() {
68. public void run() {
69. new Coder();
70. }
71. });
72. }
73. }
```

# LAYOUTS - ΔΙΑΤΑΞΕΙΣ

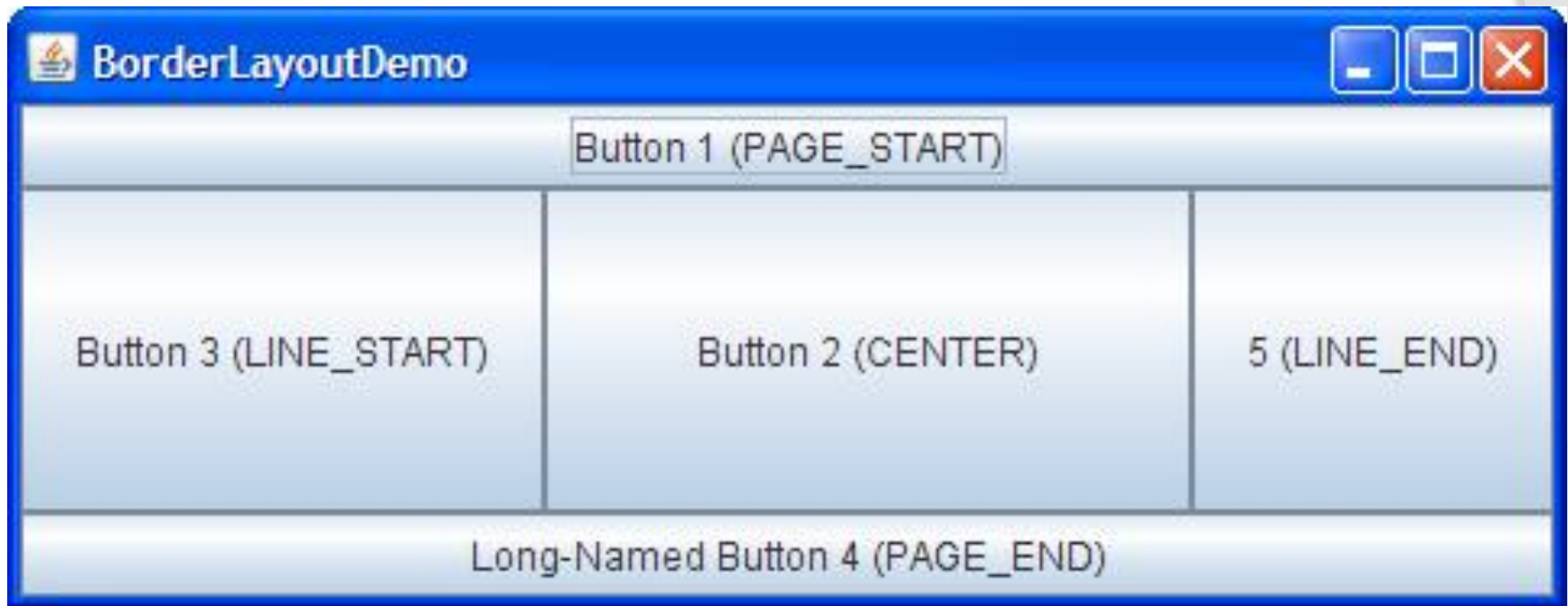
Οι διατάξεις είναι ο τρόπος που τοποθετούνται τα συστατικά σε ένα υποδοχέα.

Σημαντικότερες διατάξεις είναι οι παρακάτω:

- ◉ BorderLayout
- ◉ BoxLayout
- ◉ CardLayout
- ◉ FlowLayout
- ◉ GridBagLayout
- ◉ GridLayout

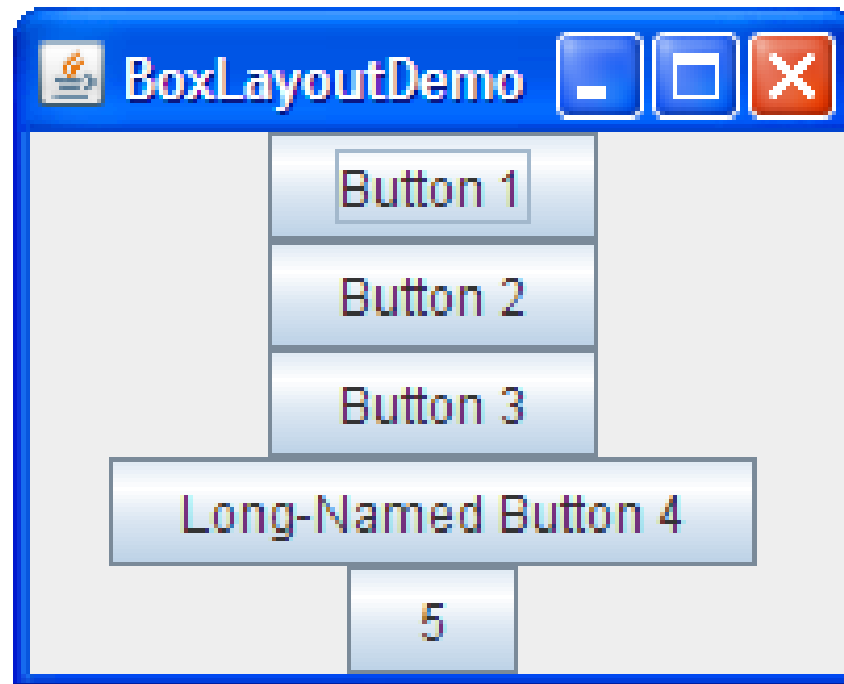
# BORDER LAYOUT

Τοποθετεί τα συστατικά σε πέντε περιοχές: top, bottom, left, right, και center.



# BOX LAYOUT

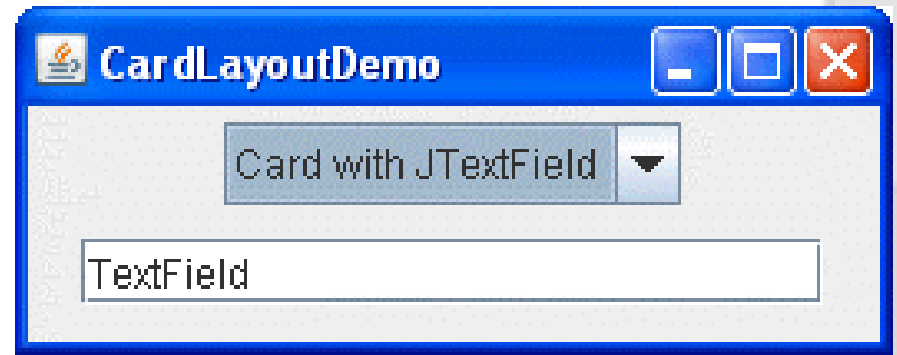
Τοποθετεί τα συστατικά σε μία στήλη.



# CARD LAYOUT

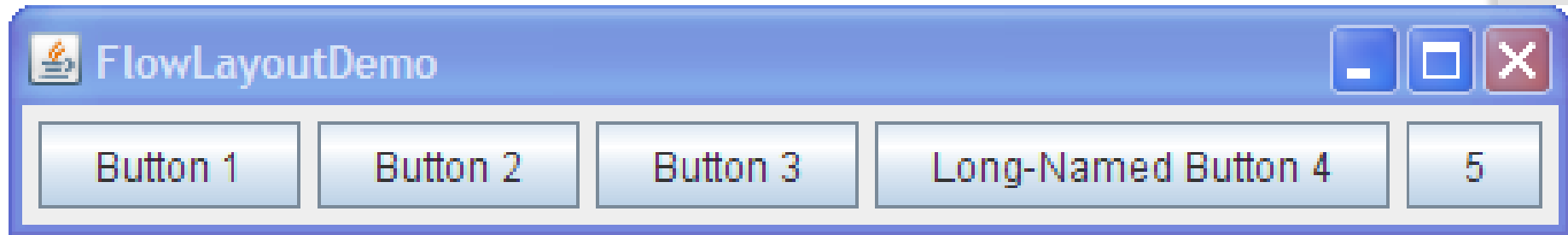
Δίνει τη δυνατότητα να τοποθετούνται τα συστατικά σε καρτέλες. Συνήθως, χρησιμοποιείται με ένα combo box για τη μεταφορά από καρτέλα σε καρτέλα.

Παρόμοιο αποτέλεσμα δημιουργεί ο υποδοχέας tabbed pane, που έχει ευκολότερη χρήση.



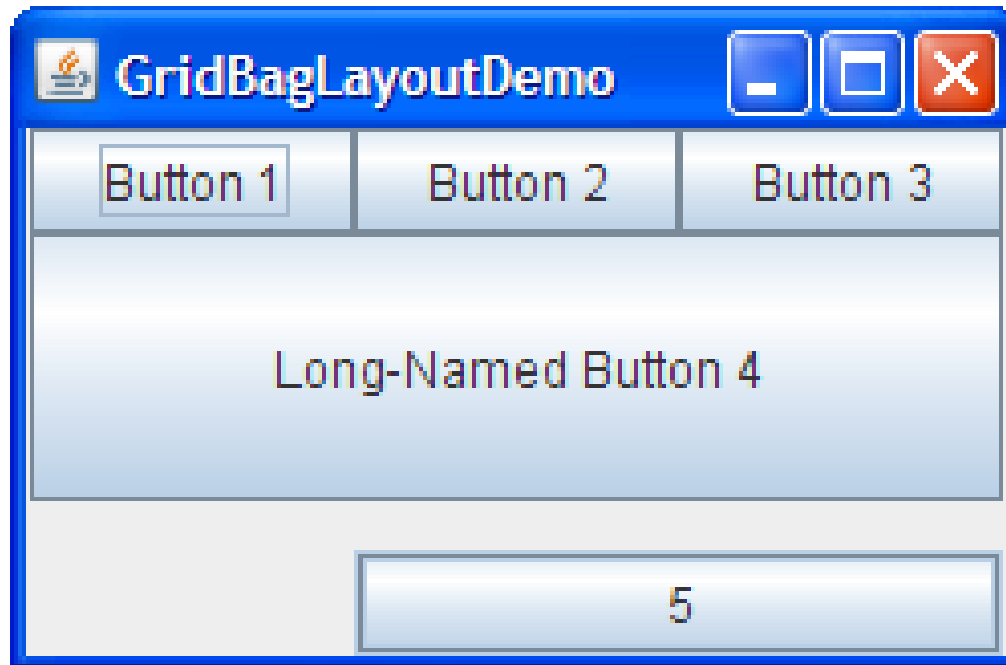
# FLOW LAYOUT

Τοποθετεί τα συστατικά σε μία σειρά. Αν δεν χωράνε τοποθετούνται στην επόμενη σειρά.



# GRID BAG LAYOUT

Τοποθετεί τα συστατικά σε πίνακα. Δίνει τη δυνατότητα για μεταβλητό πλάτος και ύψος στηλών και γραμμών. Επίσης, ένα κελί μπορεί να επεκτείνεται σε περισσότερα από ένα κελιά.





# GRID LAYOUT

Τοποθετεί τα συστατικά σε πίνακα χωρίς τις υπόλοιπες δυνατότητες του grid bag layout.



επόμενη ενότητα

# JAVA SWING

- ΕΤΙΚΕΤΕΣ
- ΚΟΥΜΠΙΑ
- ΟΡΙΑ (BORDERS)